



ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

GRADO EN INGENIERÍA INFORMÁTICA

Curso Académico 2018/2019

Trabajo Fin de Grado

**DESARROLLO DEL PENSAMIENTO COMPUTACIONAL EN NIÑOS A TRAVÉS
DE APLICACIONES: REDISEÑO Y MEJORA DE APLICACIÓN TABLET Y
CONVERSIÓN A APLICACIÓN WEB**

Autor: Andrea Rodríguez Laborda

Directores: Raquel Hijón Neira

ÍNDICE

RESUMEN	6
ABSTRACT	7
1. INTRODUCCIÓN	8
1.1 DESCRIPCIÓN DEL PROYECTO	8
1.2 OBJETIVOS	9
1.3 MOTIVACIONES	9
2. ESTADO DEL ARTE.....	10
2.1 JUEGOS FÍSICOS	10
2.1.1 Cubetto	10
2.1.2 Bee-bot	11
2.1.3 Robot Mouse.....	11
2.1.4 TrueTrue.....	12
2.2 JUEGOS DIGITALES.....	13
2.2.1 Scratch Jr	13
2.2.2 Hopscotch: Coding for Kids	13
2.2.3 Daisy the Dinosaur	14
2.2.4 Cargo-Bot.....	15
2.2.5 Kodable	15
2.2.6 Light-bot.....	16
2.3 TABLA COMPARATIVA.....	16
3. HERRAMIENTAS UTILIZADAS	18
3.1 LENGUAJES	18
3.1.1 HTML (<i>HyperText Markup Language</i>).....	18
3.1.2 CSS (<i>Cascading Style Sheets</i>).....	18
3.1.3 Javascript	18
3.1.4 PHP (<i>Hypertext Preprocessor</i>).....	18
3.1.5 SQL (<i>Structured Query Language</i>).....	19
3.2 SOFTWARE	19
3.2.1 Netbeans 8.2.....	19
3.2.2 XAMPP 7.2.11.....	19
3.2.3 Bootstrap.....	19
3.2.4 Maria DB	19
3.2.5 Tool Paint SAI	20
3.2.6 Adobe XD	20
4. DESCRIPCIÓN INFORMÁTICA.....	20
4.1 ANÁLISIS.....	20
4.1.1 <i>Visión general de la aplicación</i>	20
4.1.2 <i>Planificación</i>	21
4.1.3 <i>Especificación de requisitos</i>	23
4.1.3.1 Requisitos funcionales.....	23
4.1.3.2 Requisitos no funcionales	24
4.1.4 <i>Diagramas y casos de uso</i>	24

4.2 DISEÑO.....	30
4.2.1 Estructura básica de la aplicación.....	30
4.2.1.1 Pantalla de selección de idioma	30
4.2.1.2 Menú principal	31
4.2.1.3 Elementos comunes de los juegos	32
4.2.1.4 Ventanas modales.....	33
.....	34
4.2.2 Pantalla de los juegos.....	34
4.2.2.1 Algoritmos/Algorithms.....	34
4.2.2.2 Autómatas/Automatons.....	35
4.2.2.3 Bucles/Loops	36
4.2.2.4 Condicionales/Conditionals.....	37
4.2.2.5 Instrucciones/Instructions	37
4.2.2.6 Pasos/Steps.....	38
4.2.2.7 Patrones/Pattens	39
4.2.2.8 Cubitto	40
4.3 IMPLEMENTACIÓN.....	41
4.3.1 Funcionamiento general.....	41
4.3.2 Web vs Móvil.....	41
4.3.4 Conexión con la base de datos.....	43
4.3.5 Aleatoriedad	45
4.3.5.1 Excepciones	46
4.3.6 Funcionamiento de los juegos.....	47
4.3.7 Retos en la codificación de los juegos	50
4.3.7.1 Inicialización de instrucciones	50
4.3.7.2 Gestión del video Bucles.....	51
4.3.7.3 Posicionamiento de Cubitto y control de movimientos	52
4.4 PRUEBAS.....	54
4.4.1 Prueba General	54
4.4.2 Pruebas de usuarios.....	55
5. CONCLUSIONES	56
6. TRABAJOS FUTUROS.....	57
6.1 CONVERSIÓN A PROGRESSIVE WEB APP.....	57
6.2 PERMITIR LA INCORPORACIÓN DE NUEVAS VARIANTES	57
6.3 INCLUIR NIVELES DE DIFICULTAD	57
BIBLIOGRAFÍA	58

ÍNDICE DE FIGURAS

FIGURA 1: TABLERO, MANDO Y PIEZAS DE CUBETTO	10
FIGURA 2: ROBOT PROGRAMABLE BEE-BOT.....	11
FIGURA 3: ROBOT PROGRAMABLE ROBOT MOUSE.....	12
FIGURA 4: ROBOT PROGRAMABLE TRUE TRUE.....	12
FIGURA 5: VENTANA DEL JUEGO SCRATCH JR.....	13
FIGURA 6: VENTANA DEL JUEGO HOPSCOTCH	14
FIGURA 7: VENTANA JUEGO DAISY THE DINOSAUR.....	14
FIGURA 8: VENTANA DEL JUEGO CARGO-BOT.....	15
FIGURA 9: VENTANA DEL JUEGO KODABLE.....	15
FIGURA 10: VENTANA DEL JUEGO LIGHTBOT.....	16
FIGURA 11: DIAGRAMA DEL MODELO EN CASCADA CON SUBPROYECTOS	22
FIGURA 12: DIAGRAMA DE ACTIVIDAD.....	25
FIGURA 13: PANTALLA DE SELECCIÓN DE IDIOMA WEB.....	30
FIGURA 14: PANTALLA DE SELECCIÓN DE IDIOMAS TABLET	30
FIGURA 15: VENTANA MENÚ PRINCIPAL TABLET.....	31
FIGURA 16: VENTANA MENÚ PRINCIPAL PROTOTIPO.....	31
FIGURA 17: VENTANA MENÚ PRINCIPAL WEB.....	32
FIGURA 18: VENTANA MENÚ PRINCIPAL VERSIÓN MÓVIL	32
FIGURA 19: VENTANA MODAL EN CASO DE ACIERTO	33
FIGURA 20: VENTANA MODAL EN CASO DE FALLO.....	33
FIGURA 21: VENTANA DE AYUDA 1.....	33
FIGURA 22: VENTANA DE AYUDA 2.....	34
FIGURA 23: VENTANA ALGORITMOS TABLET	34
FIGURA 24: VENTANA ALGORITMOS WEB	35
FIGURA 25: VENTANA AUTÓMATAS WEB	35
FIGURA 26: VENTANA AUTÓMATAS TABLET	35
FIGURA 27: VENTANA BUCLES TABLET.....	36
FIGURA 28: VENTANA BUCLES WEB	36
FIGURA 29: VENTANA CONDICIONALES TABLET.....	37
FIGURA 30: VENTANA CONDICIONALES WEB.....	37
FIGURA 31: VENTANA INSTRUCCIONES TABLET	37
FIGURA 32: VENTANA INSTRUCCIONES WEB.....	38
FIGURA 33: VENTANA PASOS TABLET	38
FIGURA 34: VENTANA PASOS WEB.....	38
FIGURA 35: VENTANA PATRONES TABLET.....	39
FIGURA 36: VENTANA PATRONES WEB.....	39
FIGURA 37: TABLERO, MANDO, PIEZAS Y CUBETTO	40
FIGURA 38: VENTANA CUBITTO WEB.....	40

FIGURA 39: VERSIÓN DE LA APLICACIÓN PARA DISPOSITIVOS MEDIANOS Y GRANDES	42
FIGURA 40: VERSIÓN DE UN JUEGO PARA DISPOSITIVOS PEQUEÑOS.....	42
FIGURA 41: EJEMPLO DE VENTANA CON POSITION ABSOLUTE	43
FIGURA 42: CÓDIGO FUNCIÓN PARA ABRIR LA CONEXION CON LA BASE DE DATOS.....	43
FIGURA 43: CÓDIGO FUNCIÓN PARA CERRAR LA BASE DE DATOS	44
FIGURA 44: CÓDIGO PARA OBTENER EL ÍNDICE DE LA PRÓXIMA VARIANTE QUE SE MOSTRARÁ	44
FIGURA 45: EJEMPLO DE PASO DE VARIABLES PHP A VARAIBLES JAVASCRIPT	44
FIGURA 46: PRIMERA VERSIÓN DEL ALGORITMO DE ALEATORIEDAD	45
FIGURA 47: ALGORITMO FINAL DE ALEATORIEDAD (PARTE 1).....	45
FIGURA 48: ALGORITMO FINAL DE ALEATORIEDAD (PARTE 2).....	46
FIGURA 49: ALGORITMO FINAL DE ALEATORIEDAD (PARTE 3).....	46
FIGURA 50: EJEMPLO DE FUNCIÓN SELECT	47
FIGURA 51: : EJEMPLO DE FUNCIÓN DESELECT	47
FIGURA 52: EJEMPLO DE FUNCIÓN REORGANIZE	48
FIGURA 53: EJEMPLO DE FUNCIÓN ISCORRECT	48
FIGURA 54: EJEMPLO DE FUNCIÓN CERRAR.....	49
FIGURA 55: EJEMPLO DE FUNCIÓN VIDEO	50
FIGURA 56: FUNCIÓN PRINCIPAL BUCLES (PARTE 1).....	51
FIGURA 57: FUNCIÓN PRINCIPAL BUCLES (PARTE 2).....	52
FIGURA 58: FUNCIÓN INICIALIZACIÓN CUBITTO.....	53
FIGURA 59: FRAGMENTO FUNCIÓN MOVES.....	54

ÍNDICE DE TABLAS

TABLA 1: COMPARATIVA DE LOS JUEGOS EVALUADOS	17
TABLA 2: REQUISITOS FUNCIONALES	23
TABLA 3: REQUISITOS NO FUNCIONALES.....	24
TABLA 4: CASO DE USO 1. SELECCIONAR EL IDIOMA.....	26
TABLA 5: CASO DE USO 2. JUGAR AL JUEGO DE BUCLES/LOOPS.....	26
TABLA 6: CASO DE USO 3. INTENTAR RESOLVER UN JUEGO CON CONSULTA A BASE DE DATOS.....	27
TABLA 7: CASO DE USO 4. INTENTAR RESOLVER EL JUEGO DE INSTRUCCIONES/INSTRUCTIONS	28
TABLA 8: CASO DE USO 5. INTENTAR RESOLVER EL JUEGO PATRONES/PATTERNS	29

RESUMEN

El pensamiento computacional consiste en la resolución de problemas mediante conceptos informáticos. Esta metodología se está empezando a implantar en escuelas y centros de enseñanza debido al gran auge de las nuevas tecnologías (Wing, 2006). Hoy en día existen numerosas aplicaciones y juegos para desarrollar este tipo de pensamiento, sobre todo para los más pequeños. Sin embargo, la mayoría de ellos no son fácilmente adquiribles para todos los centros educativos, algunos juegos son demasiado caros para que los centros puedan adquirirlos.

Por ello, el objetivo de este proyecto es la creación de una aplicación web educativa para enseñar el pensamiento computacional para niños. La elección de esta plataforma se debe al avance que ha tenido el desarrollo web, lo que permite que una aplicación desarrollada para ser utilizada en un ordenador puede ser accedida también desde un móvil o Tablet siempre y cuando tengan conexión a internet. La aplicación consta de 8 juegos que se centran en diversas técnicas informáticas de resolución de problemas:

- **Bucles:** permite comprender el concepto de bucle y repetición. Para ello, el usuario debe seleccionar el número de veces que quiere que un personaje determinado realice una acción.
- **Pasos:** permite comprender el concepto de orden e instrucciones. En él, el usuario debe completar el poste que se le muestra en el orden correcto.
- **Instrucciones:** enseña a seguir una serie de pasos predefinidos. Estos pasos consisten en una serie de flechas (arriba, abajo, izquierda y derecha) que muestran los movimientos que haría el personaje perdido, y el usuario deberá seleccionar la casilla donde este personaje acabaría.
- **Algoritmos:** ayuda al usuario a entender la automatización de soluciones y la división de problemas en pasos más simples. Para ello, el usuario debe ordenar una serie de pasos que corresponden a una acción.
- **Patrones:** muestran como se puede conseguir distintos resultados cambiando algún elemento pero manteniendo el mismo contexto. En este juego el usuario debe elegir el color, pegatina, carrocería y ruedas que corresponden al coche que se muestra.
- **Automatas:** permite aprender a analizar e identificar la solución adecuada para cada caso. Para cumplir este objetivo, el usuario debe ordenar 3 colores correspondientes a los caminos, mostrados en un mapa, que se necesitan para llegar del origen al destino mostrados.
- **Condicionales:** permite empezar a comprender el concepto de if/else. Para ello, el usuario debe seleccionar 3 prendas que cumplan la condición propuesta y 3 prendas que no.
- **Cubitto:** supone un primer acercamiento a la programación. En este juego, el usuario debe guiar al personaje principal Cubitto a llegar a su destino seleccionando los pasos que debe seguir.

La aplicación permite jugar tanto en español como en inglés mejorando así la experiencia de juego.

ABSTRACT

Computational thinking consists in solving problems through computer concepts. This methodology is beginning to be implemented in schools and educational centers due to the great boom of new technologies (Wing, 2006). Today there are numerous applications and games to develop this type of thinking, especially for the little ones. However, most of them are not easily available for all schools, some games are too expensive for schools to acquire them.

Therefore, the objective of this project is the creation of an educational web application to teach computer thinking for children. The choice of this platform is due to the progress made by the web development, which allows an application developed to be used on a computer can also be accessed from a mobile or tablet as long as they have an internet connection. The application consists of 8 games that focus on various computer techniques for problem solving:

- **Loops:** allows you to understand the concept of looping and repetition. To do this, the user must select the number of times he wants a certain character to perform an action.
- **Steps:** allows you to understand the concept of order and instructions. In it, the user must complete the dessert that is shown in the correct order.
- **Instructions:** teach you to follow a series of predefined steps. These steps consist of a series of arrows (up, down, left and right) that show the movements that would make the lost character, and the user must select the box where this character would end.
- **Algorithms:** helps the user to understand the automation of solutions and the division of problems in simpler steps. For this, the user must order a series of steps that correspond to an action.
- **Patterns:** show how you can get different results by changing some element but keeping the same context. In this game the user must choose the color, sticker, body and wheels that correspond to the car shown.
- **Automatons:** allows you to learn how to analyse and identify the appropriate solution for each case. To achieve this goal, the user must order 3 colors corresponding to the roads, shown on a map, that are needed to arrive from the origin to the destination shown.
- **Conditional:** allows you to begin to understand the concept of if / else. To do this, the user must select 3 items that meet the proposed condition and 3 items that do not.
- **Cubitto:** it is a first approach to programming. In this game, the user must guide the Cubitto main character to reach their destination by selecting the steps to follow.

The application allows you to play in both Spanish and English thus improving the gaming experience.

1. INTRODUCCIÓN

En este apartado se procederá a ofrecer una breve descripción del proyecto. También se explicarán los objetivos principales del mismo y los motivos por los que se ha decidido llevarlo a cabo.

1.1 Descripción del proyecto

El pensamiento computacional es un proceso de pensamiento que involucra la formulación de problema y su resolución en la forma en que una computadora, humana o máquina, la llevaría a cabo efectivamente (Wing, 2006).

Hoy en día se está implantando en numerosos centros escolares. Es por ello por lo que existen numerosos juegos y aplicaciones que ayudan a desarrollar estas capacidades, sobre todo para los más pequeños. Sin embargo, estos juegos suelen ser inaccesibles por la mayoría de los centros debido a su gran costo y a la necesidad de materiales específicos como Tablets para la utilización de dichos juegos.

Por estos motivos, se va a desarrollar una aplicación educativa web que será multiplataforma, es decir, puede ser accedida mediante un ordenador, Tablet o smartphone siempre que tenga conexión a internet.

Para el diseño de este proyecto, se va a partir de una aplicación para Tablet ya existente la cual sigue la estructuración propuesta por Christian Brackmann en su conferencia “Pensamiento Computacional Unplugged para niños: Enseñanza de la computación sin el protagonista” (Brackmann, 2017). Dicha propuesta consistía en la división del pensamiento computacional en una serie de métodos de resolución de problemas basados en conceptos informáticos. Estos conceptos son los siguientes: Algoritmos, Autómatas, Bucles, Condicionales, Instrucciones, Pasos y Patrones.

La aplicación para Tablet ya existente, conocida como “Computational Thinking” o “Compthink” y de la cual este proyecto heredará el nombre, consiste en 7 juegos que se centran en la resolución de problemas basándose cada uno en un concepto antes descrito. Esta estructura se continuará en el proyecto, sin embargo, se añadirán funcionalidades nuevas y se mejorarán las funcionalidades actuales. Este proyecto se va a desarrollar de tal modo que permita la independencia del dispositivo que se utilice, por lo que debe tener diseño adaptativo o “responsive”, es decir, que se adapte a cualquier tamaño de pantalla. Además, para mejorar su funcionalidad, el juego podrá ser utilizado tanto en español como en inglés.

1.2 Objetivos

El objetivo principal del proyecto es la creación de una aplicación web educativa para enseñar los conceptos del pensamiento computacional a niños de entre 5 y 7 años, aunque podrían usar la aplicación niños menores de 5 años si son ayudados por un adulto.

- El juego se centrará en la enseñanza de conceptos básicos de programación como bucles, instrucciones, autómatas, condicionales, pasos, patrones y algoritmos, y no en la enseñanza de un lenguaje de programación en sí.
- La aplicación podrá ser utilizada desde cualquier dispositivo con conexión a internet sin importar el tamaño de la pantalla, a fin de poder llegar a un público más amplio.
- Los juegos deben ser lo más simples posible para que la atención esté centrada en el aprendizaje de los juegos y no en el juego en sí.
- Los juegos deben contener poco texto y las explicaciones deben ser lo más claras posibles de cara a que los juegos sean entendidos con facilidad, debido a la corta edad del usuario de la aplicación.
- Los colores empleados deben ser atractivos para los niños.
- La aplicación debe contener botones grandes para permitir el uso a usuarios sin motricidad fina, como niños pequeños.
- La aplicación debe estar disponible para todo el mundo, por ello, se debe subir a un servidor.

1.3 Motivaciones

La enseñanza y el aprendizaje de la programación y la tecnología está a la orden del día en los colegios e institutos. Sin embargo, en la mayoría de los casos se centran en la enseñanza de la robótica y la programación por bloques en vez del aprendizaje de los pensamientos computacionales.

Por tanto, se considera muy importante la enseñanza de estos métodos de resolución de problema durante el aprendizaje de los niños.

Además, pese a ser una rama de la informática en ciernes, no existen apenas aplicaciones web que cumplan esta finalidad, lo que deja un gran espacio de crecimiento.

2. ESTADO DEL ARTE

A continuación, se procede a analizar algunas de los juegos educativos disponibles para el aprendizaje del pensamiento computacional en niños, centrándose en los juegos que tengan como objetivo un público de máximo 7 años que es en la franja de edad de interés en nuestro proyecto. Posteriormente, se realizará una comparación entre estos juegos y aplicaciones descritas para ver sus características principales.

2.1 Juegos físicos

En este apartado se describen los juegos que no necesitan de ningún elemento electrónico para ser utilizado.

2.1.1 Cubetto

En este primer juego, el niño debe programar a Cubetto (ver Figura 1), el personaje principal, para que se desplace por el mapa y así seguir los pasos de la historia. En la figura 1 vemos el tablero, el mando y las piezas que se necesitan para programar a Cubetto, así como el mismo Cubetto. Esta programación se hace mediante fichas que se colocan en el mando para controlar a Cubetto. El juego no solo ayuda a comprender el concepto de instrucciones y pasos, sino que también incorpora el concepto de bucles. Sin embargo, supone un gran coste.



Figura 1: Tablero, mando y piezas de Cubetto

Fue creado en 2013 por *Primo Toys* (Primo Toys, 2019) y diseñado sobre la metodología Montessori, metodología en la cual el aprendizaje se realiza en un ambiente preparado y todos los elementos que allí se encuentran tienen una finalidad en el aprendizaje del niño (Fundación Montessori, 2018).

Al tener un diseño centrado en los niños de 3 a 5 años, la curva de aprendizaje es muy rápida y los niños comienzan a dejar cometer fallos muy rápidamente (Gabriela Caguana Anzoategui, 2017).

2.1.2 Bee-bot

Bee-bot (Terrapin, 2019), es un robot fácilmente programable que recibe órdenes básicas como adelante, atrás, izquierda y derecha a través de los botones que se encuentran en la parte superior (ver Figura 2). Ha sido desarrollado para su uso a partir de los 3 años, por lo que no se necesita saber leer para poder usarlo.

En la figura 2 podemos ver a Bee-bot y podemos apreciar los botones con los que se controla su movimiento.

También se dispone de una aplicación para poder controlarlo desde un móvil o Tablet. No dispone de ningún tipo de complemento como tablero o niveles, pero existen una gran cantidad de recursos en internet (Robots educativos, 2019).



Figura 2: Robot programable Bee-bot

2.1.3 Robot Mouse

Robot Mouse (ver Figura 3) es un set de robótica para niños a partir de 4 años, que consiste en la programación de los movimientos de un robot-ratón a través de un laberinto (Learning Resources, 2019). En la figura 3 aparece el Robot Mouse junto con los accesorios que contiene el set de robótica. Estos accesorios consisten en una serie de paneles verdes que son utilizados para formar un camino y unas barreras o puentes que se deben colocar según indica la tarjeta de nivel. Mediante los botones superiores de robot, el usuario debe programarlo para que llegue hasta el queso. Además de estas tarjetas de instrucciones por niveles, el usuario también puede crear sus propios caminos y niveles. Cuenta con una serie de cartas para que los niños puedan planificar sus órdenes antes de llevarlas a cabo (Robots educativos, 2019).



Figura 3: Robot programable Robot Mouse

2.1.4 TrueTrue

TrueTrue (TrueTrue, 2019) es un robot programable creado en Corea del Sur y que cuenta con numerosas funciones (ver Figura 4). Como vemos en la figura 4, permite la programación mediante tarjetas y por bloques. También cuenta con funcionalidades de seguimiento de líneas y sensores de proximidad que evitan que se choque. Cuenta con su propia aplicación desde la que también se puede programar. Aunque está diseñado principal mente para alumnos de primer ciclo de primaria (6-7 años), su fácil uso permite que hasta los más pequeños puedan usarlo. Gracias a que permite la programación por tarjetas, no es necesario saber leer para utilizarlo (i-Screammedia, 2017).



Figura 4: robot programable True True

2.2 Juegos digitales

En este apartado se describen los juegos que no necesitan de ningún elemento electrónico para ser utilizado.

2.2.1 Scratch Jr.

Scratch Jr. (ScratchJr, 2019) es la versión para los más pequeños del gran conocido juego de Scratch (ver Figura 5). Permite la programación de personajes y escenarios mediante programación por bloques. Cuentan con poco texto para que el saber leer no sea imprescindible. Además, permite un alto grado de customización (Acosta, 2017).

En la figura 5, podemos ver como se está programando al personaje principal para que realice una serie de acciones (representadas abajo) en un escenario que el propio usuario ha creado.



Figura 5: Ventana del juego Scratch Jr.

2.2.2 Hopscotch: Coding for Kids

Hopscotch (Hopscotch Technologies, 2019) es una aplicación para Tablet gratuita y en castellano que se centra en la creación de videojuegos y animaciones (ver Figura 6). Para ello, se centra en la programación por bloques, con la cual el niño crear y programar distintos objetos. Lo que la distingue de otros juegos parecidos, es que permite programar basándose en los sensores de la Tablet como el giroscopio (Fernández, 2014).

En la figura 6 se observa una imagen de Hopscotch, en la parte izquierda se ven las posibles acciones que pueden realizar los personajes organizados por color según el tipo y a la derecha se puede ver los diferentes personajes y la secuencia de acciones que se van a realizar cuando se pulse el botón de "play" situado a la derecha del todo.

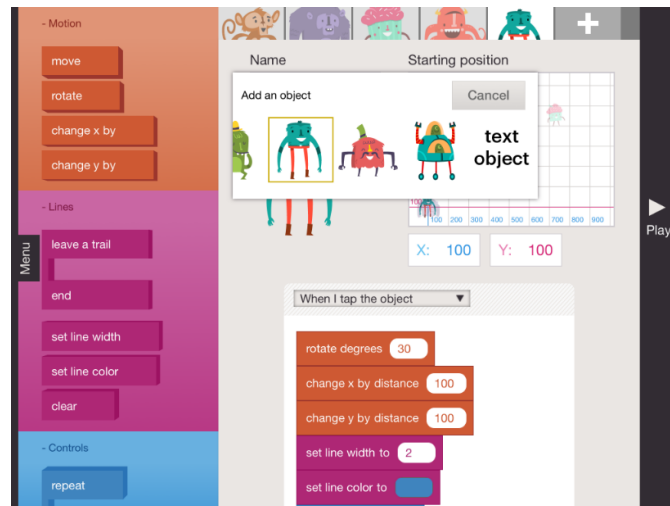


Figura 6: Ventana del juego Hopscotch

2.2.3 Daisy the Dinosaur

De los mismos creadores que la anterior (Hopscotch Technologies, 2019) y disponible solo en inglés, esta aplicación para Tablet (Common Sense Media, 2019) se centra en la programación por bloques para programar acciones del personaje principal (ver Figura 7). Cuenta con dos modos de juego, un modo por niveles, donde el niño debe programar las órdenes para cumplir un objetivo; y modo libre, donde el niño puede desplegar su creatividad (Fernández, 2014).

En la figura 7 podemos ver ala izquierda los pasos que puede realizar Daisy con el nombre “Commands”. En la parte superior podemos ver las ordenes que el usuario va a dar a Daisy, etiquetadas con “program” y, por último, en la parte inferior vemos a Daisy en el escenario, etiquetado con “stage”.

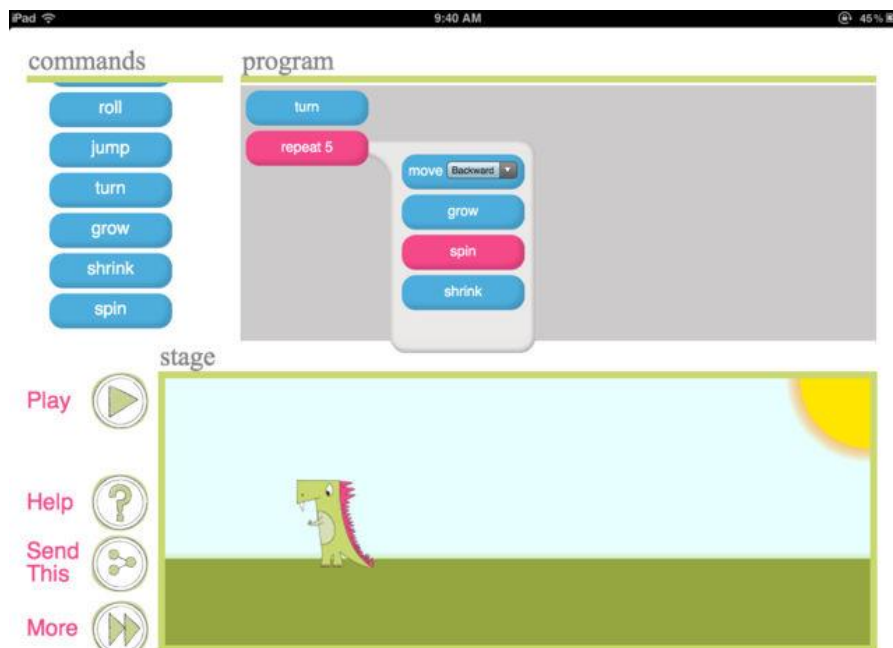


Figura 7: Ventana juego Daisy the Dinosaur

2.2.4 Cargo-Bot

En este juego (Two Lives Left, 2019) creado por Rui Viana, el usuario, mediante instrucciones simples como derecha, izquierda y abajo, debe ayudar a un robot a ordenar unas cajas siguiendo un patrón (ver Figura 8). No solo ayuda a los niños a aprender a programar, sino que ayuda a entender los conceptos de patrones y bucles. El juego consta de 36 niveles ordenados por dificultad (Fernández, 2014).

En la figura 8 se puede ver como se está resolviendo un nivel del juego. Las ordenes que está siguiendo el personaje, así como las posibles ordenes que puede ejecutar se encuentran en la parte derecha de la figura mientras que en la parte izquierda se muestra el estado al que se debe llegar y el estado actual.

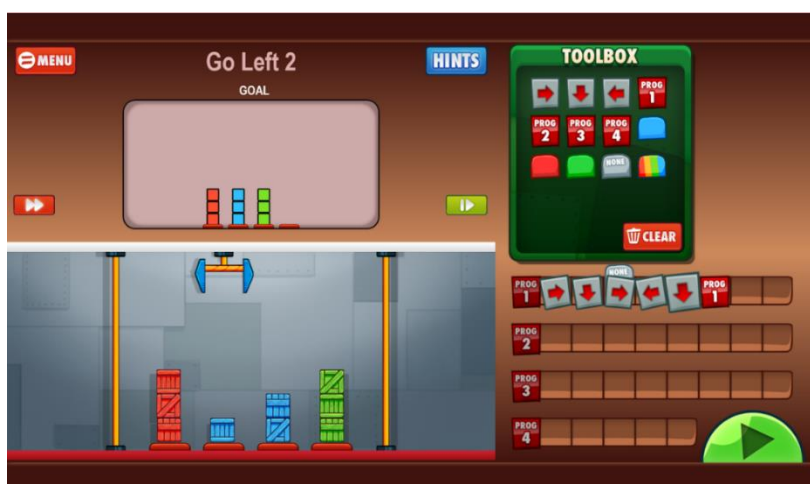


Figura 8: Ventana del juego Cargo-bot

2.2.5 Kodable

Recomendado para niños de 5 años o más, esta aplicación semigratuita (Kodable, 2019) permite aprender los conceptos básicos de programación (ver Figura 9). Como en el caso anterior, este juego deja de lado la programación por bloques para centrarse en programación mediante instrucciones sencillas (Fernández, 2014).

En la figura 9 se puede ver como se está desarrollando un nivel. En la parte superior vemos las ordenes que se le ha dado a Kodable (el personaje circular verde que vemos) para que llegue hasta el final recogiendo las monedas del camino.

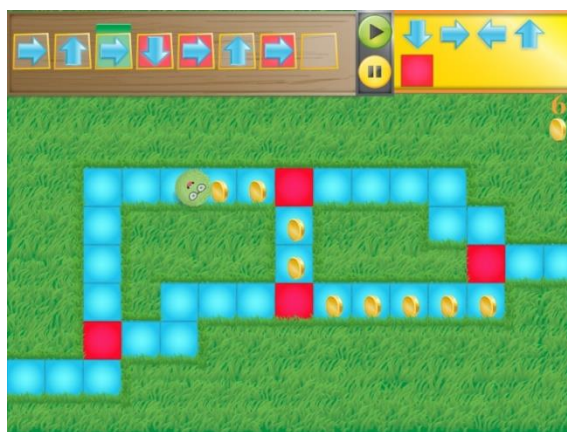


Figura 9: Ventana del juego Kodable

2.2.6 Light-bot

Otra aplicación semigratuita y con un funcionamiento muy parecido al anterior es Lightbot (LightBot Inc, 2017). Cuenta con instrucciones sencillas de dirección y algunas más complejas como salto, bucles y condicionales (ver Figura 10). El usuario debe guiar a un robot hasta su destino y encender la luz (Fernández, 2014).

En la figura 10 podemos un nivel del juego. En la parte central vemos el escenario y el robot que tenemos que programar, en la parte inferior las posibles acciones que le podemos indicar al robot que realice, y a la derecha las ordenes que le han sido asignadas.

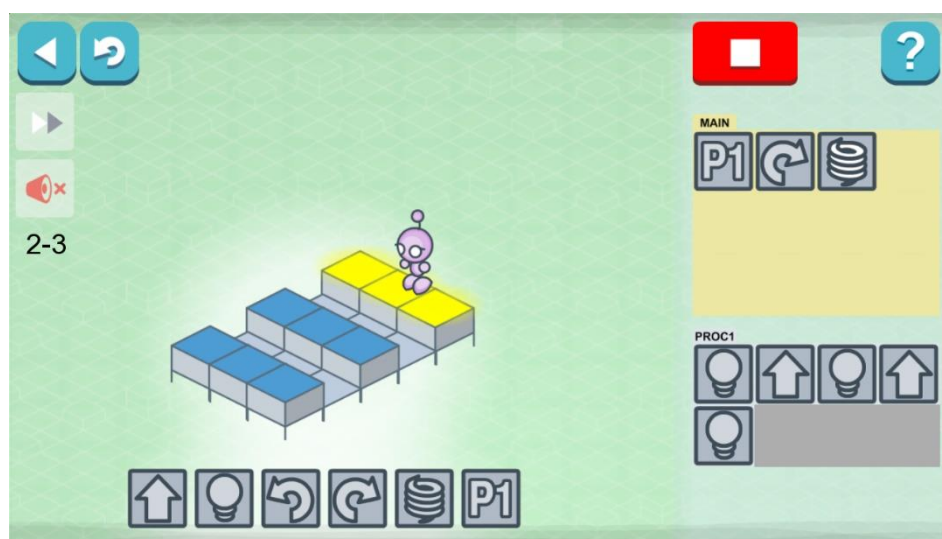


Figura 10: Ventana del juego Lightbot

























































2.3 Tabla comparativa

Tras una evaluación de todas las aplicaciones y juegos que se han encontrado, se ha procedido a una comparación de sus funcionalidades y características principales. En cuanto a los juegos físicos, la mayoría no se han podido probar debido a su elevado coste. Tras esta recopilación de información, se ha decidido comparar las características relacionadas con los objetivos del proyecto como idioma, si es multiplataforma, si se necesita tener capacidad de lectura y los paradigmas de programación que enseña, o con las características más relevantes de los juegos como en que plataforma están hechos (física o digital) y la capacidad para ser customizables, es decir, que puedan ser manipuladas para crear contenido propio.

En la siguiente comparativa (ver Tabla 1), se ve como no muchos de los juegos permiten jugar en varios idiomas y, en cuanto a la accesibilidad, aunque algunos de los juegos físicos poseen una aplicación para poder jugar con móvil o Tablet, ninguno de ellos permite jugar desde múltiples dispositivos (algunas aplicaciones solo están disponibles para Android y otras solo para Apple) y ninguno de ellos permite jugar en el ordenador.

Acerca de la necesidad de saber leer, al ser todos juegos destinados a los más pequeños, la mayoría no necesitan de esta capacidad. Un buen número de los juegos descritos permiten a los usuarios el juego libre, pudiendo crear ellos mismos sus propios niveles. Por último, aunque todos cumplen con el objetivo de enseñar el concepto de instrucciones a los usuarios, pocos cubren más paradigmas de la programación como bucles, patrones o condicionales.

Tabla 1: Comparativa de los juegos evaluados

Herramienta	Idioma	Físico/Digital	Multiplataforma	Necesita saber leer	Customizable	Paradigmas de programación que enseña			
						Bucles	Instrucciones	Patrones	Condicionales
Scratch Jr.		Digital							
Hopscotch		Digital							
Daisy the Dinosaur		Digital							
Cargo-Bot		Digital							
Kodable		Digital							
Light-bot		Digital							
Cubetto		Físico							
Bee-bot	-	Ambos							
Robot Mouse		Físico							
TrueTrue		Ambos							

3. HERRAMIENTAS UTILIZADAS

Se procede a explicar en este apartado las herramientas utilizadas y su papel desempeñado en el proyecto. Para ello, este apartado se divide en dos partes: lenguajes y software.

3.1 Lenguajes

Estos son los lenguajes que se han utilizado para el desarrollo de la aplicación. La mayoría de ellos han requerido un estudio previo de cierta profundidad, pues no se abordan de forma extensa durante el grado.

3.1.1 HTML (HyperText Markup Language)

Lenguaje de marcado comúnmente usado en el desarrollo web (MDN, 2019). Proporciona la estructura principal a la web. En concreto, he utilizado la versión HTML5, la cual ha sido muy útil debido a los nuevos elementos que proporcionaba como `<audio>` y `<video>`.

3.1.2 CSS (Cascading Style Sheets)

Lenguaje de hojas de estilos que controla el aspecto de los documentos escritos en HTML (Uniwebsidad, 2019). Su función principal ha sido la de controlar el aspecto de la aplicación y adecuarla a los distintos dispositivos.

3.1.3 Javascript

Es un lenguaje de programación que permite crear contenido nuevo y dinámico. (MDN, 2019). Se ha utilizado para permitir las interacciones del usuario con los distintos elementos de cada juego, y para comprobar si la solución seleccionada era correcta.

3.1.4 PHP (Hypertext Preprocessor)

Es un lenguaje de código abierto muy utilizado en el desarrollo web debido a que puede ser incrustado en el código HTML. Este código es procesado en el servidor. (The PHP Group, 2019) En este caso, se ha usado principalmente para hacer la conexión con la base de datos y para evitar repeticiones de código para la cabecera (header) y el pie (footer). También se ha usado para la inicialización de algunos juegos (especialmente los que hacían alguna consulta a la base de datos).

3.1.5 SQL (Structured Query Language)

Lenguaje vinculado a la gestión de bases de datos relacionales que permite la consulta de información mediante operaciones matemáticas y lógicas. (Gardey, 2010) Aunque también se podría haber usado un lenguaje NoSQL (pues las tablas para los diferentes juegos no se relacionan) se ha optado por un lenguaje relacional como SQL debido a que su uso está más extendido y ya conocía su funcionamiento.

3.2 Software

Estos son los programas o softwares que se han utilizado en mayor o menor en el proyecto.

3.2.1 Netbeans 8.2

Netbeans (The Apache Software Foundation, 2019) es un entorno de desarrollo integrado (IDE, por sus siglas en inglés) de código abierto escrito en java que soporta múltiples lenguajes de programación. Además, existe un número importante de módulos para extender este entorno (Kusterer, 2019).

Se decidió utilizar este IDE debido a que trabajaba con todos los lenguajes que utiliza el proyecto y permitía añadir un módulo de depuración para PHP. Sin embargo, y aunque la utilidad de este entorno fue razón suficiente para su uso, dicho módulo de depuración no pudo ser implementado debido al alto coste de tiempo que suponía su aprendizaje.

3.2.2 XAMPP 7.2.11

XAMPP (Apache Friends, 2019) es una distribución gratuita de Apache que contiene MariaDB (MariaDB Foundation, 2019) (versión gratis de MySQL), PHP y Perl (Perl.org, 2019). Se ha optado por usar XAMPP debido a su facilidad de uso y adquisición.

3.2.3 Bootstrap

Bootstrap (Bootstrap contributors, 2019) es un framework que facilita la maquetación de sitios web, se centra exclusivamente en el front-end. (Cochran, 2012) Para utilizarlo, se ha optado por añadir el código CDN al código HTML, pues solo lo ha usado como apoyo para el CSS y la versión descargable ocupaba espacio extra innecesariamente.

3.2.4 Maria DB

Es un software de código abierto que actúa como base de datos relacional y que proporciona una interfaz SQL para acceder a los datos (MariaDB Foundation, 2019). Al ser la usada por XAMPP, no hubo demasiada posibilidad de explorar otra alternativa.

3.2.5 Tool Paint SAI

Es un programa de dibujo disponible para Windows que posee un funcionamiento fácil y un aprendizaje rápido (SYSTEMAX Software Development - PaintTool SAI, 2019). Para este proyecto, se ha utilizado SAI para realizar las imágenes y animaciones. Estas imágenes se han hecho desde 0 o a partir de imágenes con copyleft y etiquetadas como reutilizables.

3.2.6 Adobe XD

Software para crear prototipos interactivos rápidamente y de forma gratuita. También permite compartir estos prototipos con cualquier persona mediante URL (Adobe, 2019). Esta herramienta ha sido utilizada para la realización del prototipo inicial.

4. DESCRIPCIÓN INFORMÁTICA

En este capítulo se va a realizar una descripción más técnica de todo el proceso de desarrollo del proyecto, empezando por el análisis inicial y acabando por las pruebas.

4.1 Análisis

En el análisis se va a detallar el funcionamiento básico de la aplicación y los cambios realizados respecto a la versión para Tablet. También determinaré los requisitos del sistema y proporcionaré los casos de uso de la aplicación, así como la planificación del proyecto.

4.1.1 Visión general de la aplicación

La aplicación consiste en una serie de minijuegos que permiten el aprendizaje de los conceptos del pensamiento computacional para niños de infantil y primer ciclo de primaria (para los menores de cinco años es conveniente la presencia de una figura adulta, debido a la necesidad de leer).

Nuestra aplicación se basa en una de igual nombre ya existente para Tablet, a la cual le hemos añadido más funcionalidad.

Esta aplicación para Tablet contaba con los siguiente minijuegos:

- **Bucles/Loops:** permite comprender el concepto de bucle y repetición.
- **Pasos/Steps:** permite comprender el concepto de orden e instrucciones.
- **Instrucciones/instructions:** enseña a seguir una serie de pasos predefinidos.
- **Algoritmos/algorithms:** ayuda al usuario a entender la automatización de soluciones y la división de problemas en pasos más simples.
- **Patrones/patterns:** muestran como se puede conseguir distintos resultados cambiando algún elemento pero manteniendo el mismo contexto.
- **Automatas/automatons:** permite aprender a analizar e identificar la solución adecuada para cada caso.
- **Condicionales/conditionals:** permite empezar a comprender el concepto de if/else.

Además de mejorar estos juegos, se han añadido un nuevo juego llamado Cubitto, basado en el juego Cubetto (Primo Toys, 2019) antes referido en el apartado de “Estado del arte”. Este juego añade una funcionalidad totalmente nueva a este proyecto con respecto a la versión para Tablet pues sirve como primer acercamiento a la programación en sí y, además, ayuda a comprender otra forma de solucionar problemas informaticamente.

Al igual que en la aplicación para Tablet, se permite elegir idioma entre español o inglés y todos los minijuegos cuentan con opción de ayuda, por si el usuario no comprende el funcionamiento del juego; y de salir, por si quieren cambiar de juego.

Al terminar cada minijuego, saldrá una ventana modal con un mensaje que varía dependiendo de si se ha acertado o no y que permitirá cambiar de juego y reiniciar (en caso de fallo) o cambiar de pregunta si se ha acertado.

El perfil de usuario que utilizará esta aplicación será el de un niño de entre 3 y 7 años (los niños que no sepan leer necesitarían la ayuda de un adulto). No es necesario ningún tipo de conocimientos previos de informática, pues se centran en el desarrollo de pensamientos computacionales y no en la informática en sí.

4.1.2 Planificación

Antes del desarrollo del proyecto se debe decidir qué modelo de ciclo se va a utilizar. Para este proyecto se consideró la conveniencia del modelo en cascada (OpenClassrooms, 2017).

El modelo en cascada es un proceso de desarrollo secuencial, es decir, un modelo que se conforma por una serie de etapas que se realizan una después de otra.

Sin embargo, se tomó la decisión de usar una variante de dicho modelo, conocido como cascada con subproyectos. Este modelo se diferencia del clásico en que permite que algunas tareas se ejecuten en paralelo (Lopez, 2017).

Esta decisión se tomó en base a que, aunque se tenía una base sólida en la que basarse a la hora de realizar el proyecto, el cambio de plataforma exigía una creación desde cero lo que se aprovechó a su vez para el rediseño de algunos de los juegos. Esta necesidad supuso la imposibilidad de estimar cuanto tardaría el desarrollo de cada juego, pues tendrían duraciones muy dispares, y, por tanto, un desarrollo simultáneo de los mismos parecía lo más correcto.

Por tanto, en un análisis previo pude dividir el proyecto en varias fases (ver Figura 11).

En una primera etapa se debía hacer un estudio detallado y comparativo de los diferentes juegos y aplicaciones relacionados con el pensamiento computacional, así como la aplicación para Tablet en la que se basa este proyecto.

En una segunda fase, y basándose en la información recabada, se podría realizar un prototipo de lo que sería el proyecto. Esto serviría de base a la hora de realizar la implementación.

A continuación, comenzarían una serie de fases que se tendrían que realizar una vez por cada juego. Estas fases, que corresponderían con el análisis, diseño, codificación y control de cada juego se podrían hacer simultáneamente de varios juegos a la vez, pero siempre manteniendo ese orden dentro de cada juego. El control de estos juegos sería muy básico, y no necesitaría de pruebas de usuario, sino que sería una comprobación simple del funcionamiento de todos los componentes de la ventana que se realizaría personalmente.

Cuando las fases de todos los juegos hubieran terminado, se podría proceder a la integración de todos los juegos. Y, por último, se debería realizar una prueba del juego completo que sería realizado tanto por el propio desarrollados (que comprobaría el correcto funcionamiento del mismo) como por usuarios reales que determinaría si se cumplen los objetivos del proyecto.

En la figura 11 podemos ver un diagrama de cómo se realizarían estas fases siguiendo un orden correcto.

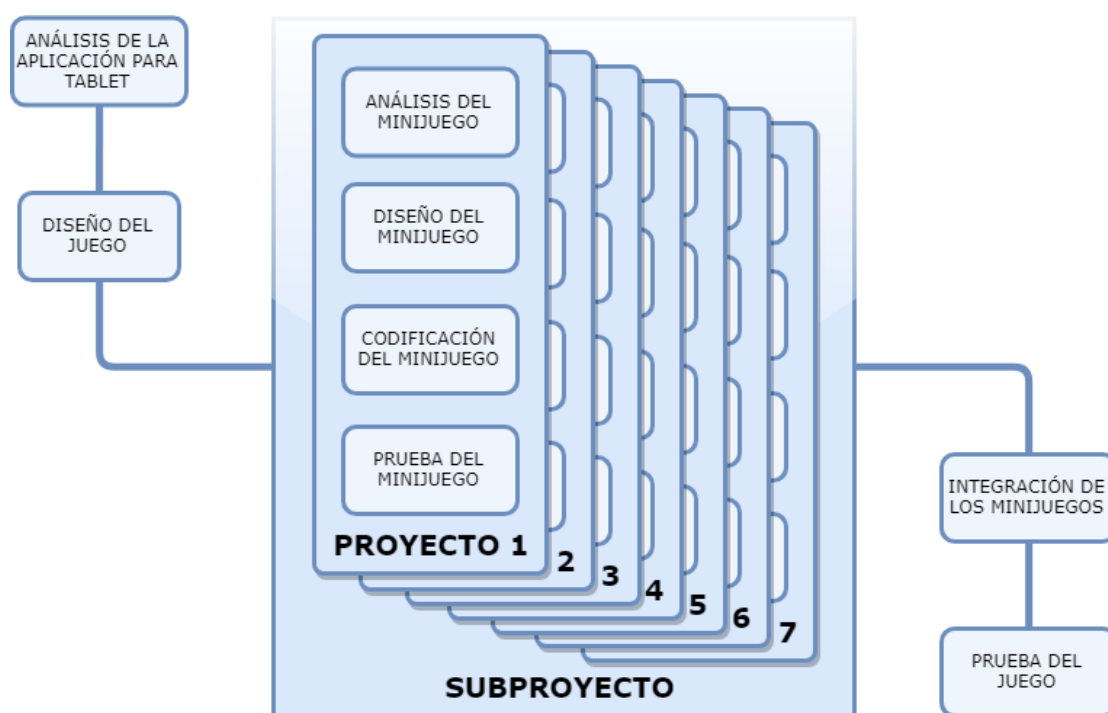


Figura 11: Diagrama del modelo en cascada con subproyectos

A estas fases hay que sumarles la fase de distribución, en el cual subimos la aplicación a un servidor para permitir el acceso a través de internet.

Cabe destacar que en este punto todavía no se había decidido la inclusión del último juego (Cubitto), por lo que en la ilustración 22 solo aparecen 7 subproyectos y no 8. Sin embargo, al haber decidido optar por este tipo de planificación, este contratiempo no tuvo ningún efecto negativo en la planificación inicial.

4.1.3 Especificación de requisitos

La especificación de requisitos es una de las fases más importantes del ciclo de vida del software. Los requisitos definen las necesidades que tiene la aplicación. Los requisitos se pueden dividir en dos categorías:

- Requisitos funcionales: son especificaciones de los servicios que precisará el sistema.
- Requisitos no funcionales: son especificaciones de las características que debe poseer el sistema.

4.1.3.1 Requisitos funcionales

En la siguiente tabla (ver tabla 2) se describen los servicios que debe cumplir la aplicación desarrollada:

Tabla 2: requisitos funcionales

NOMBRE	DESCRIPCIÓN
Elección de idioma	El sistema deberá dejar elegir al usuario en que idioma quiere jugar (español/inglés).
Elección de juego	El usuario debe poder elegir a cuál de los juegos de la aplicación quiere jugar.
Salir de un juego	La aplicación debe permitir cambiar de juego siempre que el usuario lo desee.
Ayuda	El usuario debe poder tener acceso a información de ayuda siempre que lo necesite.
Seguir jugando	El usuario debe poder seguir jugando al mismo juego tantas veces como quiera.
Cambiar de juego	El usuario debe poder cambiar el ejemplo de juego siempre que lo desee.
Feedback	El sistema deberá proporcionar feedback siempre que realice una acción.
Salir de la aplicación	El usuario puede salir de la aplicación siempre que lo desee.

4.1.3.2 Requisitos no funcionales

Una vez los explicados los servicios que debe proporcionar la aplicación, en la tabla 3 se explican las características que debe poseer:

Tabla 3: Requisitos no funcionales

NOMBRE	DESCRIPCIÓN
Disponibilidad	La aplicación debe estar siempre disponible para su uso.
Navegadores	La aplicación debe poder funcionar en cualquier navegador.
Multiplataforma	La aplicación debe verse correctamente independientemente del tamaño de pantalla del dispositivo.
Conexión a internet	La aplicación deberá funcionar correctamente siempre que tenga acceso a internet
Base de datos	El sistema debe poder acceder siempre que lo necesite a la base de datos
Facilidad de uso	La aplicación debe ser fácil de utilizar, especialmente para los usuarios objetivo (niños de hasta 7 años)
Color	La aplicación debe tener colores que atraigan la atención de los niños.
Apariencia	Como es una aplicación infantil, la aplicación debe contener poco texto, y este debe tener una tipografía atrayente para los usuarios.
Tamaño	La aplicación debe contener botones grandes para facilitar el uso al usuario objetivo.

4.1.4 Diagramas y casos de uso

Un caso de uso es una serie de pasos que definen interacciones entre el usuario y el sistema. Para un correcto caso de uso, primero se ha de describir el actor que realiza la interacción con el sistema. Para esta aplicación, el actor será un niño de entre 5 y 7 años o un niño menor de 5 años ayudado por un adulto.

A continuación, vamos a mostrar un diagrama de actividad (ver Figura 12) donde se muestre las principales acciones que el usuario puede realizar en la aplicación.

Como podemos ver en la figura 12, el juego comienza con la elección de idioma, que puede ser español o inglés.

Al seleccionar un idioma se accede al menú principal, desde el que se puede volver a la selección de idiomas. Además, desde la selección de idiomas también se puede salir del juego.

Desde el menú principal, el usuario puede acceder a cualquiera de los 8 juegos disponibles (Algoritmos, Autómatas, Bucles, Condicionales, Instrucciones, Pasos, Patrones o Cubitto). De cada uno de estos juegos se puede volver al menú principal siempre que se desee.

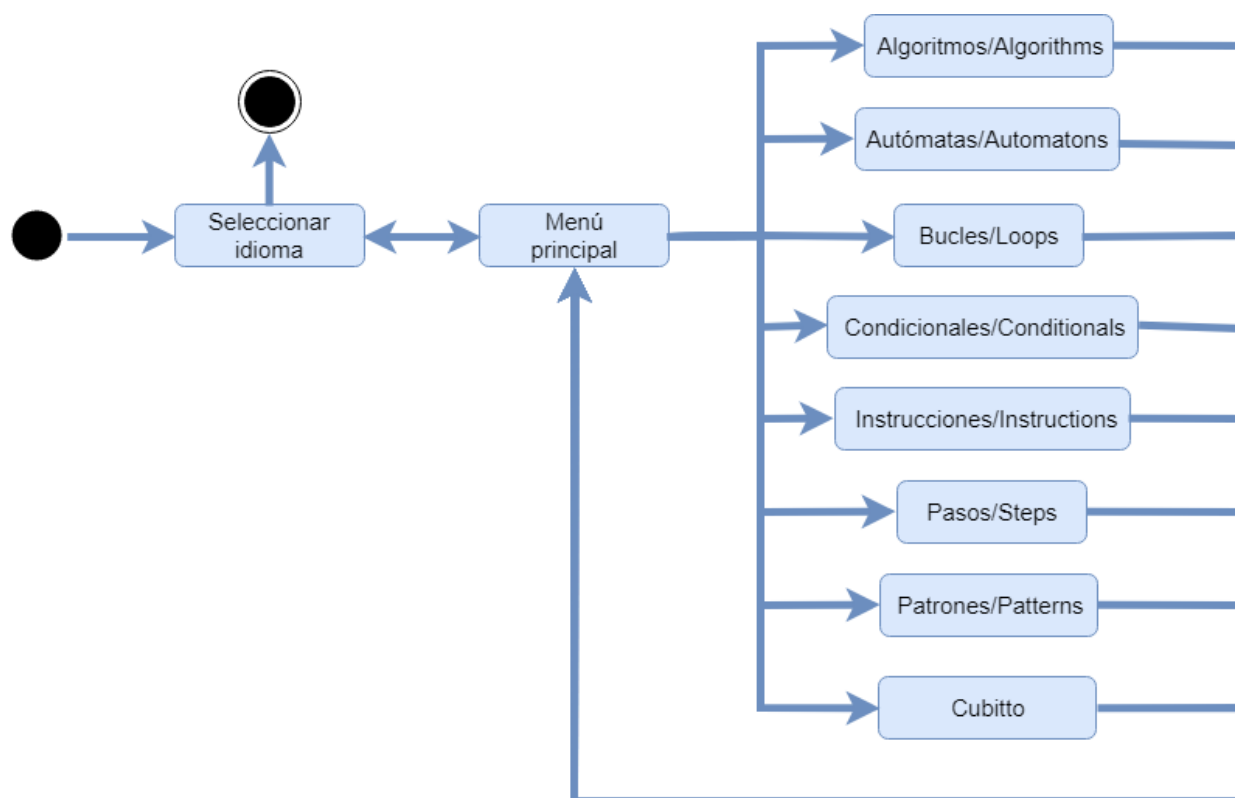


Figura 12: Diagrama de actividad

Una vez vistas las acciones principales, se procede a exponer los casos de uso que se pueden dar durante la interacción con la aplicación.

Estos casos de uso se van a centrar en las posibles acciones que podía realizar el usuario según el diagrama anterior y los flujos que puede tomar según la acción (flujo normal o flujo alternativo). También se especificará que actor realiza la acción y una descripción breve del caso de uso. Además, se incluirán las condiciones previas que necesita el caso de uso para que ocurra y las condiciones que se cumplen si termina el caso de uso.

El primer caso de uso (ver Tabla 4) corresponde a la selección del idioma por parte del usuario, lo cual se realiza al comienzo del juego o siempre que el usuario desee cambiarlo desde el menú principal.

Tabla 4: Caso de uso 1. Seleccionar el idioma.

CASO DE USO 1: SELECCIONAR IDIOMA	
Actores	Usuario
Descripción	El usuario selecciona el idioma. Puede elegir entre español o inglés.
Precondiciones	Haber entrado en el juego.
Flujo normal	<ol style="list-style-type: none"> 1. El sistema muestra la pantalla de selección de idioma. 2. El usuario selecciona un idioma pulsando en el icono de la bandera correspondiente. 3. El sistema cambia el dato de almacenamiento del idioma. 4. El sistema muestra el menú principal.
Flujo alternativo	
Postcondiciones	El juego está escrito en el idioma seleccionado por el jugador.

En el siguiente caso de uso (ver Tabla 5) describe el proceso que se produce cuando el usuario accede al juego de Bucles/Loops.

Tabla 5: Caso de uso 2. Jugar al juego de Bucles/Loops

CASO DE USO 2: JUGAR AL JUEGO DE BUCLES/LOOPS	
Actores	Usuario
Descripción	El usuario selecciona el juego de bucles.
Precondiciones	Haber entrado en el juego. Haber seleccionado el juego Bucles/Loops.
Flujo normal	<ol style="list-style-type: none"> 1. El sistema muestra al usuario el juego. 2. El usuario selecciona un número. 3. El usuario selecciona un personaje. 4. El sistema muestra un vídeo con el personaje realizando una acción las veces que indique el número. 5. Si el usuario desea salir del juego <ol style="list-style-type: none"> 5.1. El usuario pulsa el botón de salir
Flujos alternativos	5. Si el usuario desea seguir jugando vuelve al paso 2 o al 3.
Postcondiciones	Vuelve a al menú principal.

El siguiente caso de uso (ver Tabla 6) hace referencia a los juegos que hagan consulta de la base de datos (Algoritmos, Autómatas, Condicionales, Cubitto y Pasos). En este caso de uno, el usuario accede e intenta resolver uno de dichos juegos.

Tabla 6: Caso de uso 3. Intentar resolver un juego con consulta a base de datos

CASO DE USO 3: INTENTAR RESOLVER JUEGO CON CONSULTA A BASE DE DATOS	
Actores	Usuario
Descripción	El usuario selecciona el juego e intenta resolver el juego.
Precondiciones	Haber entrado en el juego. Haber seleccionado el juego desde el menú principal.
Flujo normal	<ol style="list-style-type: none"> 1. El sistema consulta la base de datos. 2. El sistema selecciona al azar una variante del juego. 3. El sistema muestra al usuario el juego que tiene que resolver. 4. El usuario selecciona una opción. 5. El sistema marca la opción seleccionada y la coloca en la posición que corresponda. 6. Si no se han marcado todas las opciones, vuelve al paso 4. 7. El sistema comprueba si la respuesta es correcta. 8. Si la respuesta es correcta <ol style="list-style-type: none"> 8.1. El sistema muestra una ventana informando al usuario de que ha acertado y le permite continuar con otra variante o salir del juego. 8.2. Si el usuario decide salir del juego <ol style="list-style-type: none"> 8.2.1.El sistema muestra el menú principal (FIN DEL CASO DE USO)
Flujos alternativos	<ol style="list-style-type: none"> 8. Si la respuesta es incorrecta <ol style="list-style-type: none"> 8.1 El sistema muestra una ventana informando al usuario de que ha fallado y le permite salir del juego o volver a intentar resolverlo. 8.2 Si el usuario selecciona salir del juego <ol style="list-style-type: none"> 8.2.1 El sistema muestra el menú principal (FIN DEL CASO DE USO) <hr/> 8.2 Si el usuario decide continuar con otra variante <ol style="list-style-type: none"> 8.2.1 El sistema consulta la base de datos. 8.2.2 El sistema selecciona una variante al azar entre las que todavía no se han mostrado. 8.2.3 Vuelve al paso 3. <hr/> 8. Si la respuesta es incorrecta <ol style="list-style-type: none"> 8.1 El sistema muestra una ventana informando al usuario de que ha fallado y le permite salir del juego o volver a intentar resolverlo. 8.2 Si el usuario decide reintentar la variante. <ol style="list-style-type: none"> 8.2.1 Vuelve al paso 3.
Postcondiciones	Vuelve a al menú principal.

El próximo caso de uso (ver Tabla 7) muestra el intento del usuario de resolver el juego de Instrucciones (en español) o Instructions (en inglés). Para este caso, el usuario debe haber seleccionado previamente el juego de instrucciones.

Tabla 7: Caso de uso 4. Intentar resolver el juego de Instrucciones/Instructions

CASO DE USO 4: INTENTAR RESOLVER EL JUEGO DE INSTRUCCIONES/INSTRUCTIONS	
Actores	Usuario
Descripción	El usuario selecciona el juego de instrucciones e intenta resolver el juego.
Precondiciones	Haber entrado en el juego. Haber seleccionado el juego Instrucciones/Instructions.
Flujo normal	<ol style="list-style-type: none"> 1. El sistema selecciona un personaje al azar. 2. El sistema selecciona al azar una casilla. 3. El sistema muestra al usuario el juego ya inicializado. 4. El usuario selecciona una casilla. 5. El sistema comprueba si la respuesta es correcta. 6. Si la respuesta es correcta <ol style="list-style-type: none"> 6.1. El sistema muestra una ventana informando al usuario de que ha acertado y le permite continuar con otra variante o salir del juego. 6.2. Si el usuario decide salir del juego <ol style="list-style-type: none"> 6.2.1. El sistema muestra el menú principal (FIN DEL CASO DE USO)
Flujos alternativos	<ol style="list-style-type: none"> 8. Si la respuesta es incorrecta <ol style="list-style-type: none"> 8.1 El sistema muestra una ventana informando al usuario de que ha fallado y le permite salir del juego o volver a intentar resolverlo. 8.2 Si el usuario selecciona salir del juego <ol style="list-style-type: none"> 8.2.1 El sistema muestra el menú principal (FIN DEL CASO DE USO) <hr/> <ol style="list-style-type: none"> 8.2 Si el usuario decide continuar con otra variante <ol style="list-style-type: none"> 8.2.1 El sistema selecciona un personaje al azar entre las que todavía no se han mostrado. 8.2.2 Vuelve al paso 2. <hr/> <ol style="list-style-type: none"> 8. Si la respuesta es incorrecta <ol style="list-style-type: none"> 8.1 El sistema muestra una ventana informando al usuario de que ha fallado y le permite salir del juego o volver a intentar resolverlo. 8.2 Si el usuario decide reintentarlo. <ol style="list-style-type: none"> 8.2.1 Vuelve al paso 3.
Postcondiciones	Vuelve a al menú principal.

En el último caso de uso (ver Tabla 8), el usuario selecciona el juego correspondiente a la resolución de problemas mediante patrones. Una vez seleccionado, el usuario trata de resolverlo.

Tabla 8: caso de uso 5. Intentar resolver el juego Patrones/Patterns

CASO DE USO 5: INTENTAR RESOLVER EL JUEGO DE PATRONES/PATTERNS	
Actores	Usuario
Descripción	El usuario selecciona el juego de patrones e intenta resolver el juego.
Precondiciones	Haber entrado en el juego. Haber seleccionado el juego Patrones/Patterns.
Flujo normal	<ol style="list-style-type: none"> 1. El sistema selecciona un coche al azar. 2. El sistema muestra al usuario el juego ya inicializado. 3. El usuario selecciona un color. 4. El usuario selecciona una pegatina. 5. El usuario selecciona una carrocería. 6. El usuario selecciona una rueda. 7. El sistema comprueba si la respuesta es correcta. 8. Si la respuesta es correcta <ol style="list-style-type: none"> 8.1. El sistema muestra una ventana informando al usuario de que ha acertado y le permite continuar con otra variante o salir del juego. 8.2. Si el usuario decide salir del juego <ol style="list-style-type: none"> 8.2.1. El sistema muestra el menú principal (FIN DEL CASO DE USO)
Flujos alternativos	<ol style="list-style-type: none"> 8. Si la respuesta es incorrecta <ol style="list-style-type: none"> 8.1 El sistema muestra una ventana informando al usuario de que ha fallado y le permite salir del juego o volver a intentar resolverlo. 8.2 Si el usuario selecciona salir del juego <ol style="list-style-type: none"> 8.2.1 El sistema muestra el menú principal (FIN DEL CASO DE USO) <hr/> <ol style="list-style-type: none"> 8.2 Si el usuario decide continuar con otra variante <ol style="list-style-type: none"> 8.2.1 Vuelve al paso 1. <hr/> <ol style="list-style-type: none"> 8. Si la respuesta es incorrecta <ol style="list-style-type: none"> 8.1 El sistema muestra una ventana informando al usuario de que ha fallado y le permite salir del juego o volver a intentar resolverlo. 8.2 Si el usuario decide reintentarlo. <ol style="list-style-type: none"> 8.2.1 Vuelve al paso 2.
Postcondiciones	Vuelve a al menú principal.

4.2 Diseño

En este apartado se detallarán las decisiones que se tomaron en lo que al diseño se refiere. Para ello, se irán comparando y explicando tanto los cambios realizados en el prototipo respecto a la aplicación para Tablet en la que se basa el proyecto, como los cambios de diseño que sufrió el producto final respecto del prototipo.

Cabe destacar que se ha tenido especial cuidado en hacer un diseño que fuera “*responsive*”, es decir, que la aplicación se vea bien independientemente del tamaño de pantalla del dispositivo.

4.2.1 Estructura básica de la aplicación

En cuanto al diseño básico de la aplicación, se ha optado por una tipografía infantil que llamara la atención del usuario y el uso de tonos pastel que cumplieran también este papel atrayente.

4.2.1.1 Pantalla de selección de idioma

En cuanto a la selección de idiomas, esta no ha sufrido prácticamente modificaciones, en la figura 13 se ve la aplicación original y en la figura 14 la resultante de este proyecto. Únicamente se le ha añadido el título de la aplicación a la pantalla. También se ha omitido el botón de salida pues al no ser una aplicación es difícil determinar a donde se debe volver.

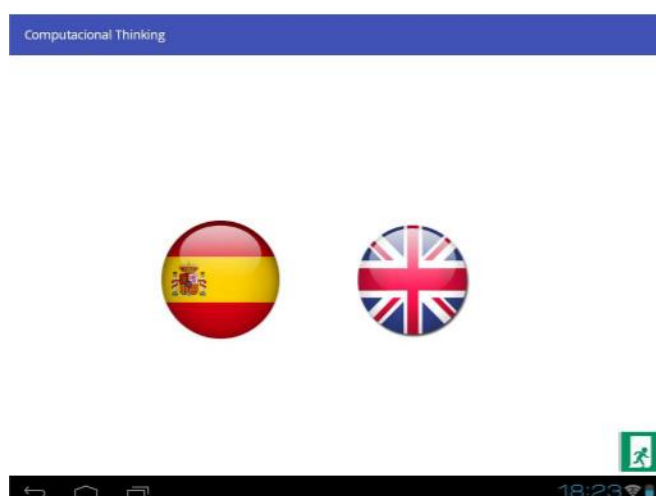


Figura 14: Pantalla de selección de idiomas Tablet



Figura 13: Pantalla de selección de idioma Web

4.2.1.2 Menú principal

La pantalla principal (figura 15) ha cambiado considerablemente respecto a su predecesora y al prototipo (figura 16), debido a la inclusión del nuevo minijuego. A raíz de la inclusión de Cubitto, el cual tenía un personaje propio, se optó por prescindir de la apariencia de biblioteca para dotar a cada juego de su protagonista que estaría asociado a un color.

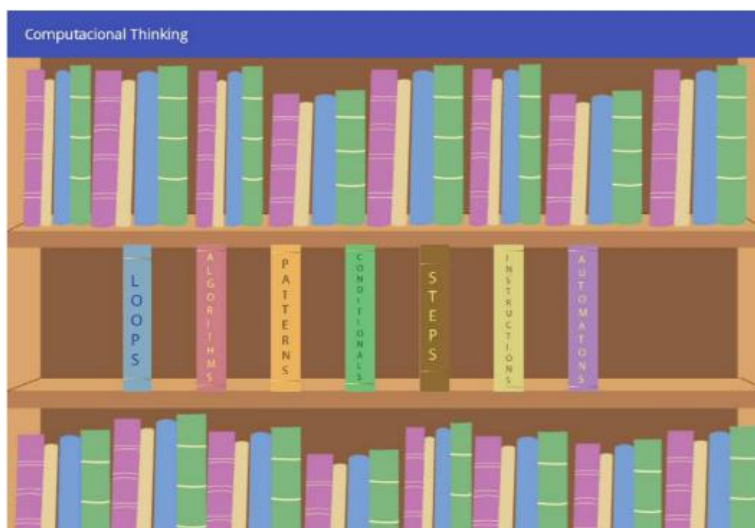


Figura 15: Ventana menú principal Tablet



Figura 16: Ventana menú principal prototipo

A continuación, podemos ver cómo ha cambiado el menú principal desde la versión para Tablet (ver Figura 15) hasta la actual (ver Figura 17). Además, a la versión final se le ha añadido un botón que permite cambiar de idioma siempre que se desee.



Figura 17: Ventana menú principal Web

Para no sobrecargar la pantalla, en el caso de que el dispositivo que utilice la aplicación sea un móvil, los personajes desaparecen dejando solo los títulos con su color correspondiente (ver Figura 18).



Figura 18: Ventana menú principal versión móvil

4.2.1.3 Elementos comunes de los juegos

Todos los juegos cuentan con botón de salir en la esquina superior izquierda y un botón de ayuda en la esquina inferior derecha. Además, los juegos que pueden llegar a ser repetitivos cuentan con otro botón que permite cambiar la variante en caso de deseirlo.

4.2.1.4 Ventanas modales

Tanto en el caso de acierto como en el caso de fallo, se muestra una ventana modal. Esta ventana modal contiene un texto indicando si se ha acertado (ver Figura 19) o no (ver Figura 20) y botones que permiten cambiar de juego, reiniciar (si se ha fallado) y seguir jugando (si se ha acertado).

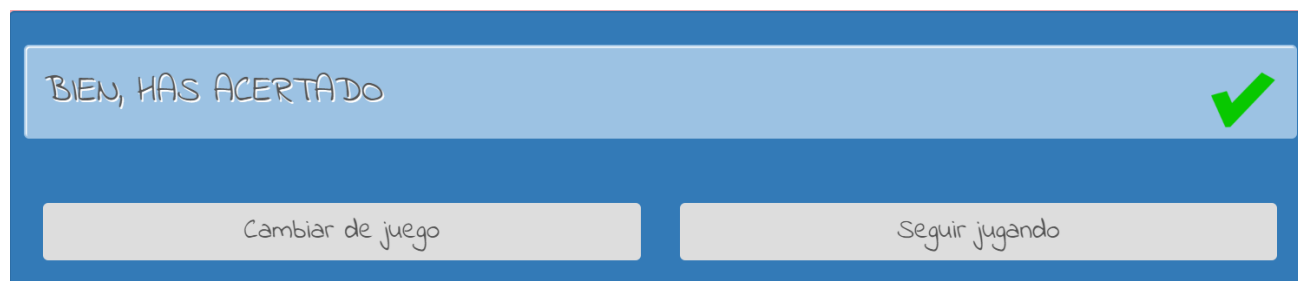


Figura 19: Ventana modal en caso de acierto



Figura 20: Ventana modal en caso de fallo

Además, se ha creado una ventana modal para mostrar la ayuda de cada juego (ver Figuras 21 y 22). Esta ayuda consiste en una breve explicación que podemos ver al comienzo de las figuras 21 y 22, dos imágenes de ayuda, como se ve en la figura 21, y un video de cómo se resuelve el juego que solo se muestra si el usuario selecciona esa opción como se puede ver en la figura 22.

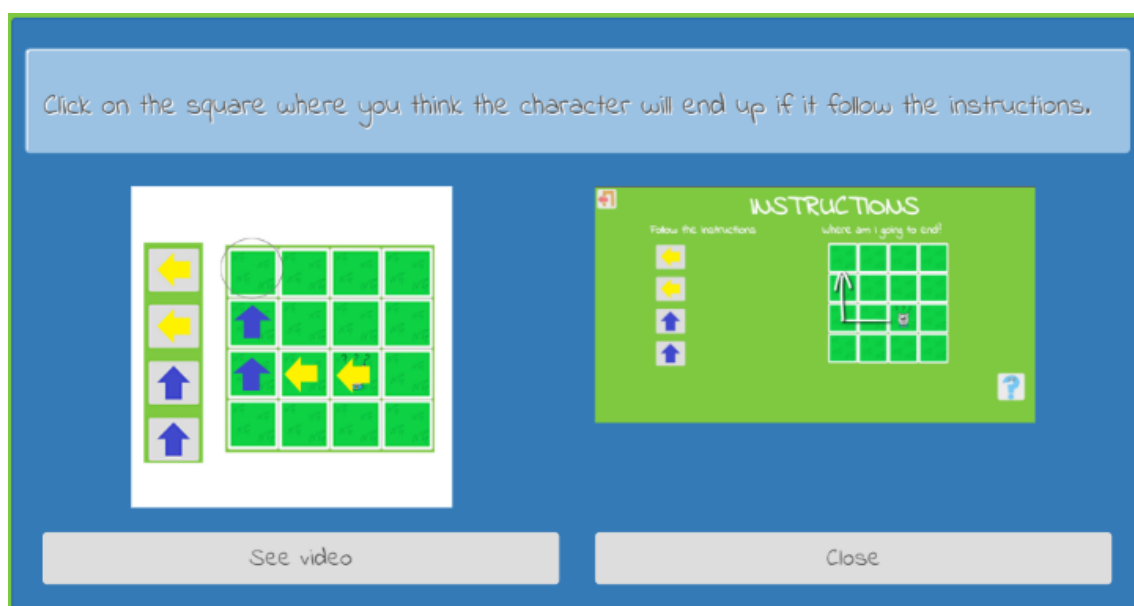


Figura 21: Ventana de ayuda 1



Figura 22: Ventana de ayuda 2

4.2.2 Pantalla de los juegos

En este apartado se explicará la apariencia de cada juego y analizará los posibles cambios entre la versión de Tablet y la final.

4.2.2.1 Algoritmos/Algorithms

Este juego consiste en una serie de acciones que el usuario debe ordenar para realizar la tarea correctamente.

Para este juego se ha decidido prescindir de los menús desplegables (ver Figura 23), que no son muy recomendables para la motricidad fina, y cambiarlos por botones, los cuales se irán colocando en el orden en que han sido pulsados (ver Figura 24). Además, se ha decidido comenzar con la comprobación de la solución justo al pulsar la última opción posible, lo que ahorra un clic y hace que el juego sea más dinámico.

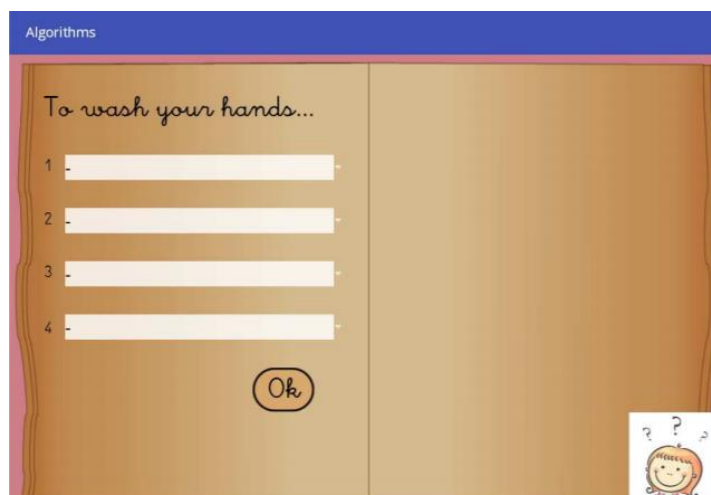


Figura 23: Ventana Algoritmos Tablet



Figura 24: Ventana Algoritmos Web

4.2.2.2 Autómatas/Automatons

Este juego conserva su apariencia original (ver Figura 25), consiste en un mapa, tres colores y un punto de origen y destino. El usuario deberá pulsar los colores, que se irán colocando en el orden en que son pulsados, a fin de conseguir formar un camino del punto de origen al de destino (ver Figura 26).

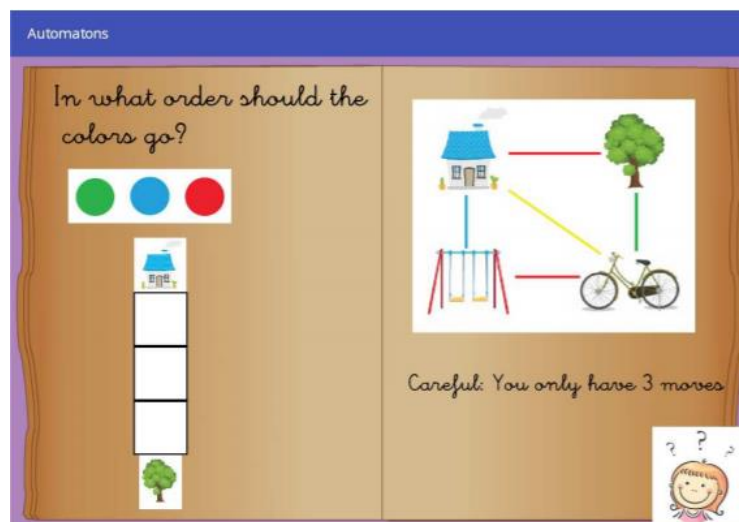


Figura 25: Ventana Autómatas Tablet

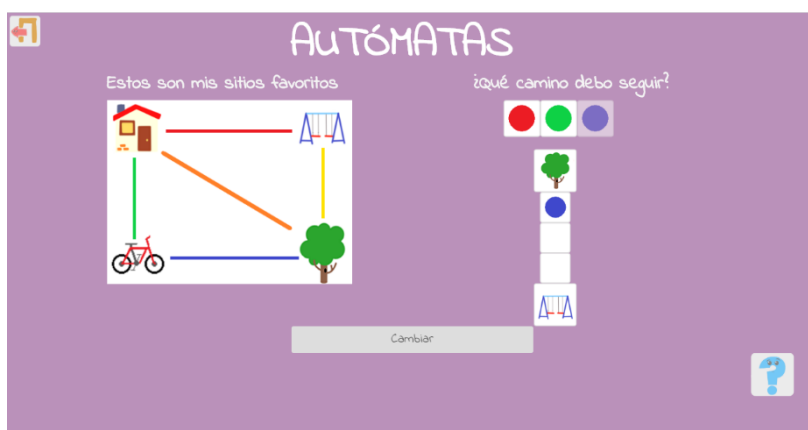


Figura 26: Ventana Autómatas Web

4.2.2.3 Bucles/Loops

El usuario debe seleccionar un número y un personaje, lo que activará la aparición de un video a la derecha donde el personaje seleccionada realiza una opción las veces del número seleccionado (ver Figura 27).

Al igual que en el caso de algoritmos, se ha prescindido del menú desplegable y nos hemos decantado por usar botones agrupados en una misma fila (Figura 28).

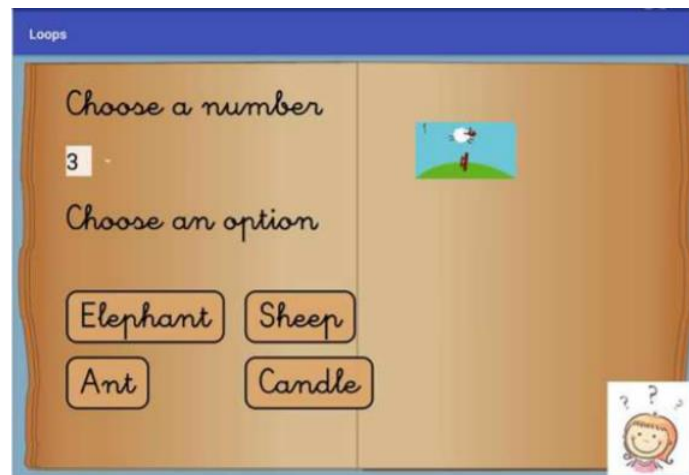


Figura 27: Ventana Bucles Tablet

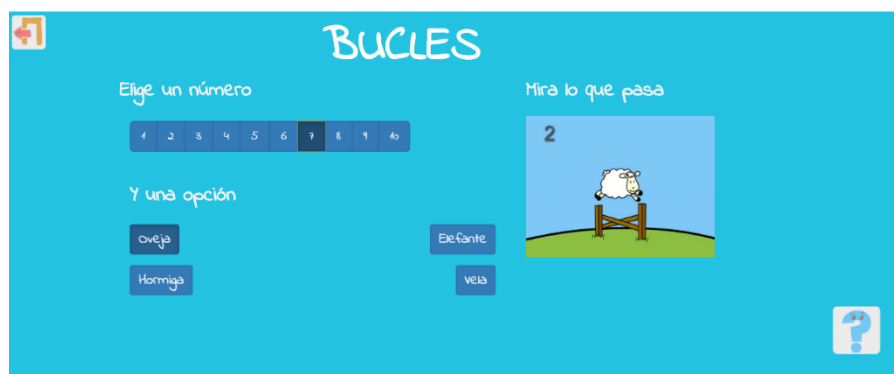


Figura 28: Ventana Bucles Web

4.2.2.4 Condicionales/Conditionals

En este juego no ha habido casi modificaciones (ver Figura 29). En él, el usuario debe seleccionar tres prendas que cumplan la condición del título, que se colocarán en los tres espacios correspondientes y tres que no la cumplan, que se colocarán en los tres espacios restantes (ver Figura 30).

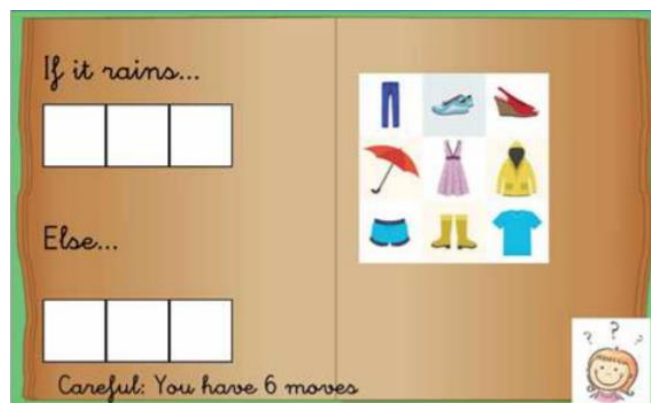


Figura 29: Ventana Condicionales Tablet



Figura 30: Ventana Condicionales Web

4.2.2.5 Instrucciones/Instructions

El usuario debe seleccionar la casilla donde acabaría el personaje si siguiera los movimientos de las flechas (arriba, abajo, izquierda o derecha) (ver Figura 31). Se ha decidido dotar a este juego de una apariencia más infantil a fin de que sea más atrayente para los niños. Además, se ha añadido una instrucción más para aumentar la dificultad (ver Figura 32).

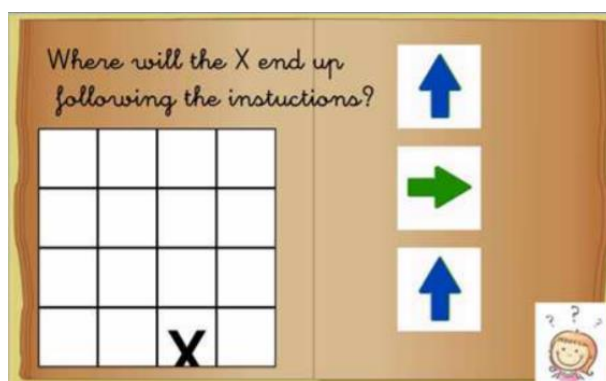


Figura 31: Ventana Instrucciones Tablet

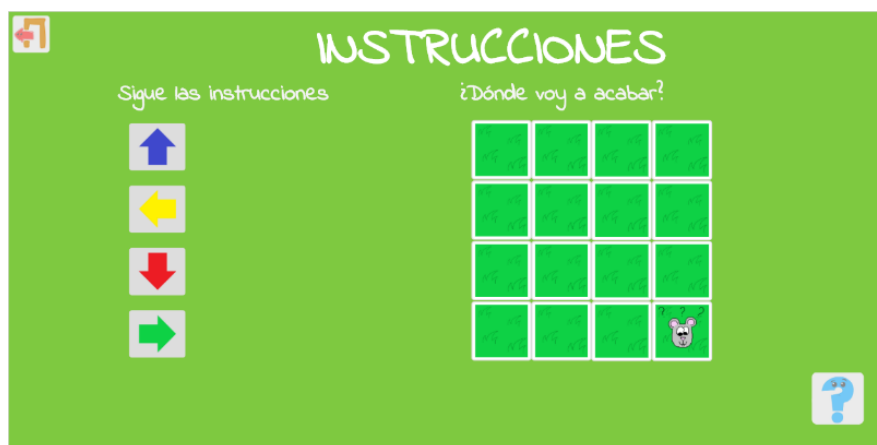


Figura 32: Ventana Instrucciones Web

4.2.2.6 Pasos/Steps

Para este juego se ha decidido cambiar totalmente la dinámica. En la versión para Tablet (ver Figura 33), el usuario debía decir que componentes faltaban de una imagen respecto de otra, se podía asimilar más a “buscar las diferencias”. En esta nueva versión (ver Figura 34), se ha decidido darle más interacción al juego por lo que el usuario no solo tendrá que decir que falta, sino que tendrá que indicarlo en el orden correcto.



Figura 34: Ventana Pasos Tablet



Figura 33: Ventana Pasos Web

4.2.2.7 Patrones/Pattens

Este juego también ha sufrido ciertas modificaciones. En un principio el usuario tenía que elegir entre chico o chica y después seleccionar un tipo de ojos, cara y boca para formar un rostro (ver Figura 35). Debido a la controversia que genera hoy en día sobre los estereotipos femeninos y masculinos, se ha decidido cambiar estas imágenes por un objeto. Además, se quería conseguir que el usuario tuviera un objetivo, y por ello se alteró la dinámica del juego.

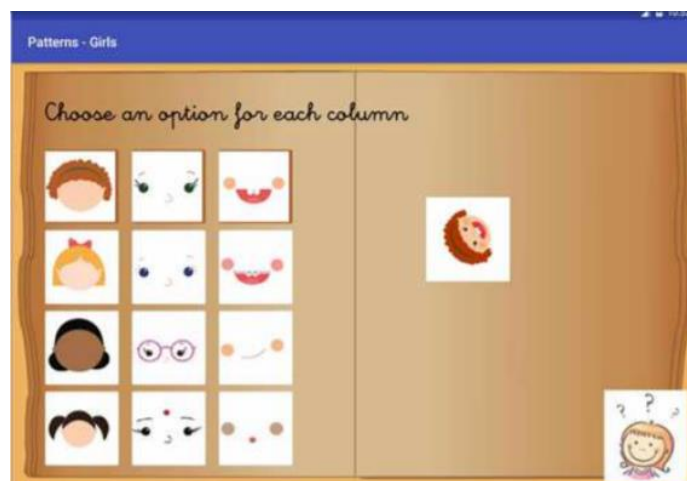


Figura 35: Ventana Patrones Tablet

Así, en la versión final (ver Figura 36), al usuario se le presenta un coche que debe construir seleccionando el color, la pegatina, la carrocería y las ruedas.

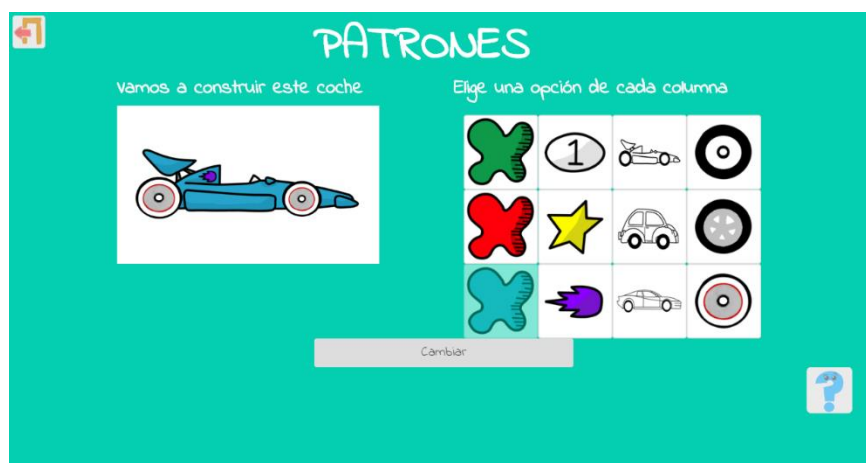


Figura 36: Ventana Patrones Web

4.2.2.8 Cubitto

Se basa en el juego físico ya existente llamado “Cubetto” (Primo Toys, 2019). En él, el usuario debe guiar al personaje principal hasta un punto determinado. Para ello dispone de un mando y de unas fichas de movimiento (ver Figura 37). Al colocar estas fichas en el mando y pulsar el botón azul, estas órdenes serán ejecutadas por el personaje.

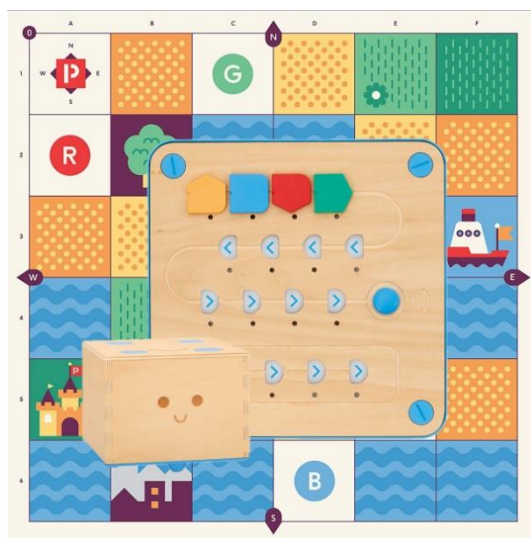


Figura 37: Tablero, mando, piezas y Cubitto

Este funcionamiento ha sido reproducido fielmente por el juego correspondiente del proyecto, permitiendo jugar a este juego físico de alto coste de forma gratuita y online. Así, escuelas y familias sin un alto poder adquisitivo podrán ser capaces de beneficiarse de este juego que permite aprender a programar de una forma fácil y entretenida.

En la figura 38 podemos ver el juego de Cubitto que hemos creado a partir de Cubetto. Así podemos ver el mando a la izquierda con las fichas ya colocadas, y el tablero con Cubitto a la derecha.



Figura 38: Ventana Cubitto Web

4.3 Implementación

Para abordar cómo se ha realizado el proceso de codificación, se analizarán los algoritmos más importantes que se han tenido que desarrollar y los mayores retos que ha supuesto el proyecto.

4.3.1 Funcionamiento general

Primero se procederá a explicar el funcionamiento del juego en cuanto a codificación se refiere.

La aplicación consta de varios archivos PHP, Javascript y CSS divididos en varias carpetas. En la carpeta principal se encuentra el archivo “index.php”, el cual contiene la codificación de la ventana principal. Además de ese archivo, el juego contiene una serie de carpetas:

- Juegos: en ella se encuentran los archivos correspondientes a cada minijuego (con terminación .php).
- Css: contiene el archivo css principal de la aplicación.
- Img: donde se almacena todas las imágenes necesarias para la aplicación.
- Templates: almacena los archivos que contienen elementos comunes de la mayoría de las ventanas.

El archivo “index” hace las veces de ventana principal y de ventana de selección de idiomas. Para saber cuál debe de mostrar utiliza la variable de sesión “idioma”. Esta variable de PHP mantiene su valor hasta que se cierre la ventana del navegador. Esto nos permite mantener esta información, aunque se cambie de página. Así, si la variable no está inicializada, el idioma aún no ha sido elegido, y por lo tanto la ventana que se muestra es la de selección de idiomas. Una vez se haya seleccionado el idioma y, por tanto, se haya inicializado el idioma, se mostrará la ventana principal.

4.3.2 Web vs Móvil

Para adaptar el juego a cualquier tamaño de pantalla se han utilizado las clases proporcionadas por Bootstrap junto con clases propias incluidas en el archivo “style.css”.

Bootstrap utiliza un sistema de cuadrícula para organizar los elementos por la página. Esta cuadrícula se organiza en 12 columnas que se pueden dividir a gusto del programador entre los contenedores que cree.

Bootstrap permite añadir varias clases de contenedores a los mismos elementos que hacen referencia a tamaños de pantalla distintos, esto favorece la organización de los elementos de forma distinta si se está utilizando un ordenador, Tablet o móvil.

Gracias a esta característica, se ha podido organizar la apariencia de la aplicación sin tener que hacer tres diseños distintos (web, Tablet y móvil).

Para dispositivos medianos y grandes (ver Figura 39), los elementos de los juegos se dividen en dos columnas mientras que para dispositivos pequeños (ver Figura 40) solo se utiliza una columna.

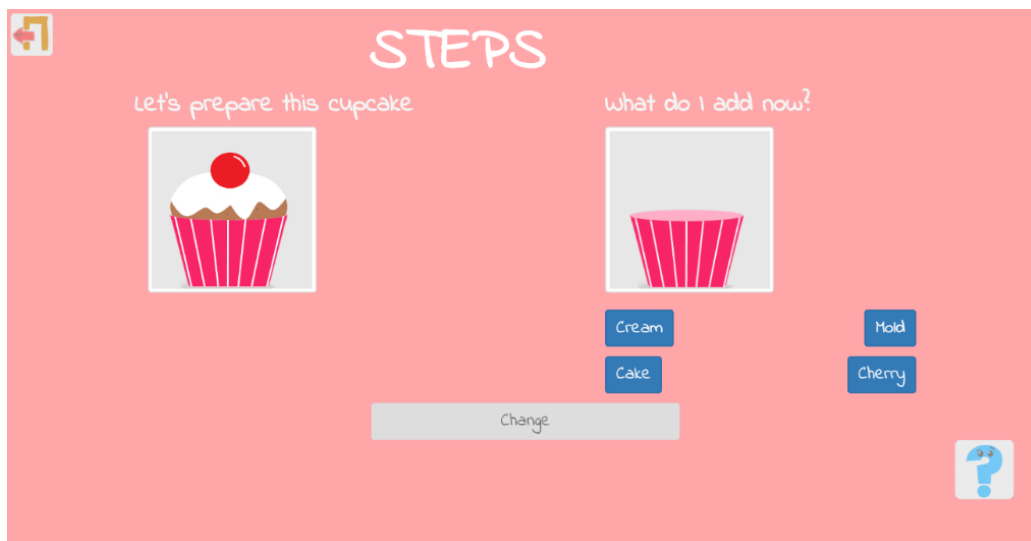


Figura 39: Versión de la aplicación para dispositivos medianos y grandes



Figura 40: Versión de un juego para dispositivos pequeños

Las ventanas modales de acierto, fallo y ayuda se encuentran en el archivo modal.php. Este código se añade a la ventana correspondiente mediante la función “include_once”. Esta función añade el código del archivo una vez al código principal. A estas ventanas se les añadió la clase “hidden” de Bootstrap, la cual deja el elemento oculto y sin ocupar espacio en la pantalla. Al producirse la condición para que la ventana aparezca esta clase es eliminada y el elemento es visible. Para que las ventanas sean visibles independientemente de los elementos de la ventana, la característica “position” de las ventanas es “absolute”, es decir, la posición de la ventana será absoluta en la pantalla, y por tanto estará fija en la ventana (ver Figura 41).

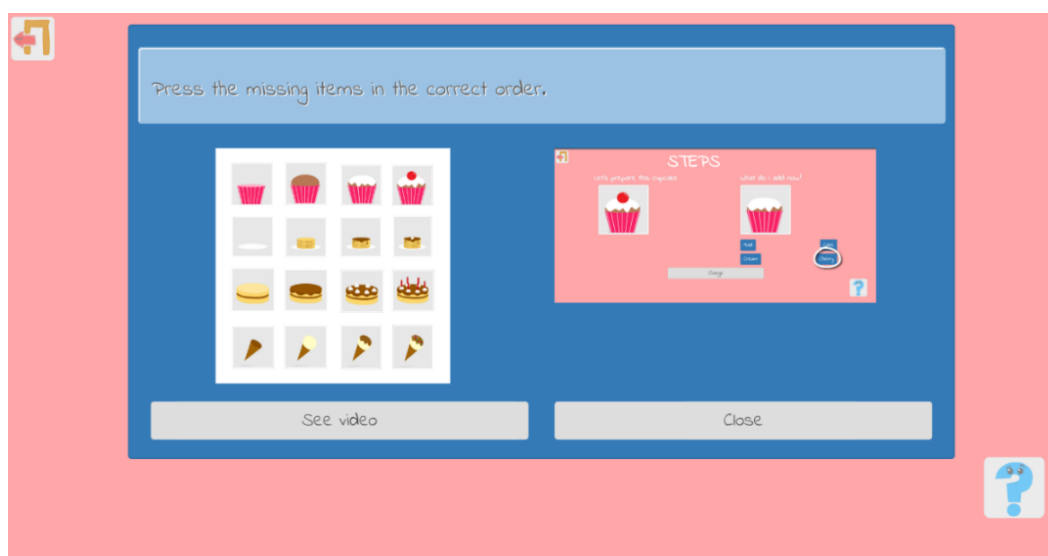


Figura 41: Ejemplo de ventana con position absolute

4.3.4 Conexión con la base de datos

Para la comunicación de la aplicación con la base de datos se ha usado el lenguaje PHP (The PHP Group, 2019). Este lenguaje, que se interpreta en el servidor, permite conectarse a la base de datos y consultar su información. Para ello, se creó un archivo con nombre “connection.php” que contiene las funciones principales para la comunicación con la base de datos. Este archivo se incluye de igual forma que las ventanas modales mediante “include_once” justo después del “header” (siempre que el juego necesite la base de datos).

Este archivo contiene tres funciones:

- connectionOpen (): para abrir la conexión con la base de datos. Como podemos ver en la figura 42, la función intenta conectarse al servidor, posteriormente establece el conjunto de caracteres predeterminado del cliente (en este caso utf8 que contiene los caracteres especiales del español). Por último, la función intenta conectarse a la base de datos en sí y la devuelve.

```
function connectionOpen() {
    $connection = mysqli_connect(SERVER_NAME, USER_NAME, PASSWORD) or die("No se ha podido conectar al servidor de Base de datos");
    mysqli_set_charset($connection, 'utf8');
    $db = mysqli_select_db($connection, DB_NAME) or die("Ups! Pues va a ser que no se ha podido conectar a la base de datos");
    return($connection);
}
```

Figura 42: Código función para abrir la conexión con la base de datos

- `connectionClose (mysqli $connection)`: para cerrar la conexión con la base de datos (ver Figura 43). Tan solo llama a la función de PHP que cierra la conexión a la base de datos que se le pasa como valor.

```
function connectionClose($connection) {
    mysqli_close($connection);
}
```

Figura 43: código función para cerrar la base de datos

- `getNext (string $title mysqli $connection)`: devuelve la siguiente variable del juego. Esta función será explicada más adelante cuando hablemos de la aleatorización de los juegos.

Cuando un juego necesita acceso a la base de datos, se llama a la primera función del archivo para abrir la conexión. Posteriormente, se llama a la última función que devuelve el índice de la próxima variante del juego que se mostrará.

Una vez obtenemos este índice, consultamos a la base de datos los datos necesarios para mostrar la variante del juego asociada a ese índice (ver Figura 44).

```
$Sql = "SELECT * FROM automatas WHERE id='$id'";
$res = mysqli_query($connection, $Sql) or die("Algo ha ido mal en la consulta a la base de datos");
while ($row = mysqli_fetch_array($res)) {
    $solution = array($row['Solucion1'], $row['Solucion2'], $row['Solucion3']);
    $Origen = $row['Origen'];
    $Destino = $row['Destino'];
    $Mapa = $row['Mapa'];
}
```

Figura 44: código para obtener el índice de la próxima variante que se mostrará

Una vez tengamos todos estos datos, es hora de cerrar la conexión con la segunda función del archivo. Es recomendable cerrar la conexión a la base de datos siempre que ya no se necesite por motivos de seguridad.

Si es necesario, estos datos se pasan a una variable de Javascript (ver Figura 45) para que se puedan pasar como parámetros a las funciones que controlan el correcto desarrollo del juego.

```
<script type="text/javascript">
    var S1 = "<?php echo $solution[0]; ?>";
    var S2 = "<?php echo $solution[1]; ?>";
    var S3 = "<?php echo $solution[2]; ?>";
</script>
```

Figura 45: Ejemplo de paso de variables PHP a variables Javascript

4.3.5 Aleatoriedad

Para que los juegos no se hicieran tediosos y repetitivos, se tuvo que diseñar un algoritmo que permitiera obtener las variables a mostrar por el juego de forma aleatoria, pero sin que se repitiesen constantemente las mismas variables.

El primer algoritmo que se ideó destacaba por su sencillez (ver Figura 46). Únicamente consultaba el máximo número de variantes del juego a la base de datos (en caso de que la tuviera) y devolvía un número aleatorio entre 1 y el máximo número de variantes. En caso de que el usuario acabara de jugar a una variante del juego y decidiera seguir jugando a otra, el sistema mandaba por POST el índice de la variante que se acababa de jugar, y al generar el nuevo juego, el sistema comprobaba que el número aleatorio generado no fuera igual que el anterior, y generaba números aleatorios hasta que no coincidiera.

```
$SqlMax = "SELECT MAX(id) FROM automatas";
$result = mysqli_query($connection, $SqlMax) or die("Algo ha ido mal en la consulta a la base de datos");

while ($row = mysqli_fetch_array($result)) {
    $max = $row[0];
}

if (isset($_POST['Otro'])) {
    do {
        $id = rand(1, $max);
    } while ($id == $_POST['Otro']);
} else {
    $id = rand(1, $max);
}
```

Figura 46: Primera versión del algoritmo de aleatoriedad

Sin embargo, con este sistema no todas las opciones se mostraban en la misma proporción, por que el usuario podía aburrirse del juego antes de haber jugado a todos los minijuegos incluidos. Además, en caso de salir del juego y volver a entrar, la información de la última variante jugada no se guardaba, por lo tanto, podía volver a aparecer la misma variante de la última vez.

Por todo esto, se decidió cambiar la forma de generar el identificador de la siguiente variante. Este nuevo algoritmo, utiliza un array de dimensión igual al número de variantes posible el cual se rellena con ceros la primera vez que se abre el juego (ver Figura 47). Este array se almacena en una variable PHP de tipo sesión, lo que permitirá mantener su valor, aunque el juego se cierre.

```
//Si aun no se ha jugado al juego
if (!isset($_SESSION[$title])) {
    $_SESSION[$title] = array();
    for ($i = 1; $i <= $max; $i++) {
        array_push($_SESSION[$title], 0);
    }
}
```

Figura 47: Algoritmo final de aleatoriedad (Parte 1)

En caso de que no sea la primera vez que se juega a este juego, el sistema comprueba si han sido mostradas todas las variantes del juego (ver Figura 48). Para ello, comprueba si hay algún campo con valor 0 en el array. De ser así, este índice todavía no se habría utilizado y, por tanto, aun quedarían variantes por mostrar. En caso contrario, todos los índices se habrían mostrado y en consiguiente se debe volver a empezar, es decir, se debe volver a inicializar el array.

```
//En caso contrario
} else {
    $sincompleto = false;
    //Comprueba que no este lleno
    foreach ($_SESSION[$title] as $indice) {
        if ($indice == 0) {
            $sincompleto = true;
        }
    }
    if (!$sincompleto) {
        $_SESSION[$title] = array();
        for ($i = 1; $i <= $max; $i++) {
            array_push($_SESSION[$title], 0);
        }
    }
}
```

Figura 48: Algoritmo final de aleatoriedad (Parte 2)

Por último, el algoritmo debe devolver el índice de la siguiente variante que se va a mostrar (ver Figura 49). Para ello genera números aleatorios entre 1 y el máximo de variantes posibles hasta que la posición correspondiente del array sea 0 (no se hubiera utilizado todavía) y el índice seleccionado no fuera igual que el anterior, para controlar que, si se el array estaba lleno y se vació, no se repita justo el mismo índice. Por último, cuando se consiga un índice que cumpla las condiciones, se pondrá el valor 1 a la posición correspondiente del array.

```
do {
    $id = rand(1, $max);
} while (1 == $_SESSION[$title][$id - 1] || (isset($_POST['Otro']) && $_POST['Otro'] == $id));
$_SESSION[$title][$id - 1] = 1;
return $id;
```

Figura 49: Algoritmo final de aleatoriedad (Parte 3)

4.3.5.1 Excepciones

En el caso del juego “Patrones”, al no necesitar base de datos, y al depender la variante de 4 factores, hay una probabilidad muy baja de que se repita exactamente la misma variante, y por tanto no necesita ningún algoritmo especial. Por tanto, para generar una nueva variante, se generan 4 números aleatorios (uno por cada característica del coche) y con estos números se llama a la imagen correspondiente.

En caso de “Cubitto”, solo se comprueba el índice de los destinos, y la posición del personaje se genera con 2 números aleatorios no importando que se repita.

Por su parte, “Bucles” no necesita ningún tipo de aleatoriedad.

4.3.6 Funcionamiento de los juegos

En este apartado se describirá la codificación básica de los juegos, en cuanto a acciones dinámicas se refiere.

Para soportar las interacciones con los usuarios, se ha utilizado el lenguaje “Javascript” (MDN, 2019), el cual permite actualizar dinámicamente la página sin recargarla, al contrario que PHP.

Se ha intentado mantener los nombres de las funciones a lo largo de los juegos siempre que mantuviera una funcionalidad común, aunque estuvieran en archivos diferentes. Estas funciones son las siguientes:

- **Selected:** se activa cuando se selecciona una opción (ver Figura 50). Puede recibir uno y dos parámetros. Cuando se activa esta función, el elemento que la accionó, que normalmente viene como parámetro, se desactiva (deshabilitándola, quitándole la clase “active”, volviéndola opaca...) y actualizan el valor de la respuesta correspondiente (cambiándole la ruta si es una imagen, el texto si es un botón...). En la mayoría de los casos, accionan otras funciones para, por ejemplo, comprobar si el resultado es correcto.

```
//Se selecciona una opción
function selected(e) {
    e.disabled=true;
    for(var i=1;i<=4;i++){
        if(document.getElementById("btn"+i).innerText=="-") {
            document.getElementById("btn"+i).innerText=e.innerText;
            if(i!=4){
                i++;
            }
            document.getElementById("btn"+i).classList.remove("hidden");
            break;
        }
    }
    changeTip();
    isCorrect();
}
```

Figura 50: Ejemplo de función select

- **Deselected:** se activa cuando se selecciona un elemento de respuesta (ver Figura 51). Recibe un parámetro. Normalmente invierte los cambios realizados por “selected”. Si al producirse los cambios las respuestas necesitan algún tipo de reorganización, llama a la función “reorganize”. También puede accionar otras funciones exclusivas de cada juego.

```
//Se deselecta una respuesta
function deselected(e) {
    var text = e.innerText;
    for(var i=1;i<=4;i++){
        if(document.getElementById("btnOp"+i).innerText===text) {
            document.getElementById("btnOp"+i).disabled=false;
        }
    }
    e.innerText="-";
    }
    reorganice();
    changeTip();
}
```

Figura 51: Ejemplo de función deselect

- **Reorganize:** se activa cuando al eliminar una respuesta, estas necesitan reagruparse (ver Figura 52). Se utiliza en las ventanas de “Algoritmos” y “Autómatas”. Tan solo comprueba que no haya huecos vacíos entre dos respuestas, y en caso afirmativo desplaza hacia arriba las respuestas en posiciones inferiores a la del espacio vacío.

```
//Si se elimina una respuesta y quedan debajo más respuestas, las respuestas posteriores adquieren la posición de sus predecesoras
function reorganize(){
  for(var i=1;i<4;i++){
    var j = i+1;
    if((document.getElementById("btn"+i).innerText=="-") && (document.getElementById("btn"+j).innerText!="-")){
      document.getElementById("btn"+i).innerText = document.getElementById("btn"+j).innerText;
      document.getElementById("btn"+j).innerText="-";
    }else if((document.getElementById("btn"+i).innerText=="-") && (document.getElementById("btn"+j).innerText=="-")){
      document.getElementById("btn"+j).classList.add("hidden");
    }
  }
}
```

Figura 52: Ejemplo de función reorganize

- **IsCorrect:** normalmente es activada por la función “selected” aunque a veces también se activa por una acción del usuario (ver Figura 53). No suele contener parámetros, pero puede contener un parámetro que normalmente se corresponde con el elemento que la activó. Normalmente esta función comprueba si se han seleccionado todas las respuestas, y en caso afirmativo comprueba si lo respondido por el usuario se corresponde con la respuesta correcta. En caso de que la respuesta es correcta, se llama a la función que muestra la ventana modal con la letra “a” como parámetro, mientras que si es incorrecta se inicia el juego con los parámetros que tenía al principio de la variante y se llama a la función que muestra la ventana modal con la letra “f” como parámetro.

```
//Comprueba si es correcto
function isCorrect(){
  if(document.getElementById("btn4").innerText!="-"){
    if((document.getElementById("btn1").innerText==S1)&&
      (document.getElementById("btn2").innerText==S2)&&
      (document.getElementById("btn3").innerText==S3)&&
      (document.getElementById("btn4").innerText==S4)){
      modal("a");
    }else{
      document.getElementById("tip").innerText = "Primero...";
      for(var i=1;i<=4;i++){
        document.getElementById("btn"+i).innerText="-";
        document.getElementById("btnOp"+i).disabled=false;
        if(i!="1"){
          document.getElementById("btn"+i).classList.add("hidden");
        }
      }
      modal("f");
    }
  }
}
```

Figura 53: Ejemplo de función isCorrect

Además de estas funciones, el archivo “app.js” contiene funciones comunes a todas las ventanas además de un objeto que contiene la explicación de cada juego en español e inglés. Las funciones son las siguientes:

- **InitAyuda:** recibe como parámetros el nombre del juego y el idioma en el que se debe mostrar el juego para inicializar el video de ayuda, el cual se denomina `ayuda_nombreDelJuego_idioma.mp4`.
- **Ayuda:** recibe el nombre del juego y el idioma, actualiza la explicación en la ventana modal de ayuda y la muestra. Para obtener esta explicación, el programa llama al objeto de Javascript que almacena esta información, selecciona la propiedad correspondiente al juego y al idioma recibidos.
- **Cerrar:** se activa cuando el usuario pulsa el botón “Cerrar” de la ventana de ayuda. Se encarga de ocultar la ventana, añadiéndole la clase “oculto”, y vuelve a inicializar la ventana si se produjo algún cambio (ver Figura 54).

```
function cerrar(v) {  
    document.getElementById('ventana-flotante-ayuda').classList.add('oculto');  
  
    document.getElementById('video-ayuda').classList.add("hidden");  
    document.getElementById('img-ayuda1').classList.remove("hidden");  
    document.getElementById('img-ayuda2').classList.remove("hidden");  
  
    if(i=="español"){  
        document.getElementById('Ver').innerHTML='Ver video';  
    }else{  
        document.getElementById('Ver').innerHTML='See video';  
    }  
}
```

Figura 54: Ejemplo de función cerrar

Modal: como comentamos anteriormente, muestra la ventana modal correspondiente dependiendo del parámetro de entrada, “a” para la ventana de acierto y “f” para la de fallo. Además, reproduce el sonido correspondiente a acierto o fallo, dependiendo de la ventana.

- **Background y Title:** se encargan de poner el fondo del color correspondiente a cada juego y añadir el título de cada juego respectivamente.

- Video: se acciona cuando el usuario selecciona la opción de ver video. La función se encarga de esconder las imágenes de ayuda y mostrar el video (ver Figura 55).

```
function video(r,i){
    document.getElementById('video-ayuda').classList.remove("hidden");
    document.getElementById('img-ayuda1').classList.add("hidden");
    document.getElementById('img-ayuda2').classList.add("hidden");
    initAyuda(r,i);
    document.getElementById('video-ayuda').play();
    if(i=="español"){
        document.getElementById('Ver').innerHTML='Repetir video';
    }else{
        document.getElementById('Ver').innerHTML='Repeat video';
    }
}
```

Figura 55: Ejemplo de función video

4.3.7 Retos en la codificación de los juegos

En este apartado se explicarán problemas o dificultades que se han tenido que enfrentar a la hora de codificar los juegos.

4.3.7.1 Inicialización de instrucciones

A la hora de generar las instrucciones que debía seguir el usuario, había que comprobar múltiples factores, y una simple inicialización de los movimientos mediante números aleatorios no sería suficiente debido a:

- Hay que comprobar que, al seguir las instrucciones, el personaje no acaba fuera de la cuadrícula.
- Hay que comprobar que las instrucciones dadas no son contradictorias (arriba seguido de abajo y viceversa e izquierda seguido de derecha y viceversa).

Para ello se creó un algoritmo que generaba una casilla inicial y cuatro números aleatorios del 1 al 4 (1 correspondía con arriba, 2 con abajo, 3 con derecha y 4 con izquierda). Además, dentro de este algoritmo se crearon varias variables:

- X: almacena la coordenada X de la posición inicial del personaje y en la que se encontraría el personaje al ir siguiendo las instrucciones.
- Y: almacena la coordenada Y de la posición inicial del personaje y en la que se encontraría el personaje al ir siguiendo las instrucciones.
- Z: almacenaba el posible próximo movimiento (1,2,3 o 4 para arriba, abajo, derecha o izquierda).
- Anterior: almacenaba el movimiento de la instrucción anterior (arriba, abajo, izquierda o derecha).
- Siguiente: almacenaba la posible coordenada que cambiaría si el personaje siguiera la instrucción que se estaba comprobando.
- Move: que será falso o verdadero en función de si la instrucción que se quería comprobar es posible o no.

El funcionamiento de este algoritmo es el siguiente:

- Se genera un número aleatorio del 1 al 4.
- Comprueba que el movimiento que corresponde al número no es el inverso al anterior.

- Actualiza la variable siguiente en función de la instrucción que se realizaría:
 - X +1 si el movimiento es hacia abajo.
 - X-1 si el movimiento es hacia arriba.
 - Y+1 si el movimiento es hacia la derecha.
 - Y-1 si el movimiento es hacia la izquierda.
- Comprueba si existe la casilla con el valor siguiente y en ese caso:
 - Actualiza el valor de la flecha.
 - Actualiza la variable x o y al valor de siguiente.
 - Pone la variable move a verdadero.
 - Actualiza el movimiento de la variable anterior.

Este proceso se realiza hasta que el movimiento se pueda realizar hasta 4 veces.

4.3.7.2 Gestión del video Bucles

Para la animación que debía mostrarse cuando el usuario seleccionaba un número y un personaje, se debía buscar un sistema para mostrarla. Tras estudiar varios formatos para la animación (GIF, video, etc.), se optó por el video debido a su bajo nivel de complejidad respecto de las demás formas.

Para ello, se utilizó el elemento <video> de HTML5 el cual permite cargar un archivo en múltiples formatos y reproducirlo. Además, este elemento permite reproducir el video desde un punto de este hasta un punto determinado lo que era realmente útil para el funcionamiento del juego.

Gracias a estas características, la función que se tuvo que crear fue bastante sencilla (ver Figura 56). Dicha función, además de gestionar la selección del número y el personaje mediante dos variables (N para el número y P para el personaje), comenzaba a reproducir el video del personaje concreto durante los segundos almacenados en la variable N.

```

if (N > 0 && P != "Z") {
  switch (P) {
    case "O":
      document.getElementById('video').src = "../img/Bucles/oveja.mp4#t=", " + ((N*2)-1)+".9";
      document.getElementById('video').play();
      break;
    case "E":
      document.getElementById('video').src = "../img/Bucles/elefante.mp4#t=", " + (N-1)+".9";
      document.getElementById('video').play();
      break;
    case "H":
      document.getElementById('video').src = "../img/Bucles/hormiga.mp4#t=", " + ((N*2)-1)+".9";
      document.getElementById('video').play();
      break;
    case "V":
      if (N != 10) {
        document.getElementById('video').src = "../img/Bucles/vela.mp4#t=", " + N + ".5";
      } else {
        document.getElementById('video').src = "../img/Bucles/vela.mp4";
      }
      document.getElementById('video').play();
      break;
  }
}

```

Figura 56: Función principal Bucles (parte 1)

En cuanto a la gestión de la selección del número y el personaje, la función recibe el nuevo número o personaje y el número o personaje que no se ha modificado. Es por esto por lo que la función comprueba si el valor de los parámetros es igual al ya almacenado por la variable o al valor por defecto y actualiza las variables en caso afirmativo (ver Figura 57).

Además, proporciona un feedback al usuario de qué botón esta pulsado añadiendo o quitando la clase active.

```
if (n > 0) {
    if (N != n && N > 0) {
        document.getElementById("btn" + N).classList.remove("active");
    }
    document.getElementById("btn" + n).classList.add("active");
}

if (p != "Z") {
    if (P != p && P != "Z") {
        document.getElementById("btn" + P).classList.remove("active");
    }
    document.getElementById("btn" + p).classList.add("active");
}

N = n;
P = p;
```

Figura 57: Función principal Bucles (parte 2)

4.3.7.3 Posicionamiento de Cubitto y control de movimientos

En el juego de Cubitto, el personaje debe moverse por la pantalla cada vez que se procese una acción. Por ello, la característica “position” del personaje debía ser “absolute” y el “z-index” ser un número alto para que siempre se coloque encima de los demás elementos.

Una vez cambiadas estas características, había que colocar al personaje en un punto concreto de la pantalla. Mediante unos pocos cálculos matemáticos esto se resolvió fácilmente.

Así, para colocar al personaje había que obtener la posición del mapa en la ventana, pues esta podría cambiar en función del tamaño de la pantalla. Para ello, se usó la función “getBoundingClientRect ()” que devuelve una serie de valores numéricos sobre el alto, ancho y distancia a ciertos puntos de un objeto desde los márgenes de la ventana.

Como el mapa es una cuadrícula de 6x6 había que dividir el alto de la imagen (conseguido mediante la función anterior) para saber cuántos píxeles habría que desplazar al personaje en cada movimiento de avance y cuanto debería medir el personaje (al ser la imagen cuadrada, con calcular la altura fue suficiente).

Antes de realizar estos cálculos, se debía generar un punto de inicio, lo cual se consiguió generando dos valores aleatorios, se debía comprobar primero que el origen fuera distinto que el destino. Una vez obtenido este dato y el número de píxeles que había que desplazar en cada movimiento, la función coloca al personaje a la altura y anchura correctas (ver Figura 58).

```

function init() {
  do {
    inicioX = Math.floor(Math.random() * 5) + 1;
    inicioY = Math.floor(Math.random() * 5) + 1;
  } while (inicioX == destinoX && inicioY == destinoY);
  x = inicioX;
  y = inicioY;
  let coords = document.getElementById("mapa").getBoundingClientRect();
  cube = coords.height / 6;
  ejeX = coords.bottom - coords.height + ((inicioX - 1) * cube);
  if(window.screen.width>768){
    ejeY = coords.left + ((inicioY - 1) * cube)-10;
  }else{
    ejeY = coords.left + ((inicioY - 1) * cube);
  }
  auxX = ejeX;
  auxY = ejeY;

  document.getElementById("cubitto").style.width = cube + "px";
  document.getElementById("cubitto").style.left = ejeY + "px";
  document.getElementById("cubitto").style.top = ejeX + "px";
  document.getElementById("width").value = cube;
}

```

Figura 58: Función inicialización Cubitto

Una vez posicionado correctamente, y cuando el usuario pulsara el botón de enviar movimientos, el personaje debía moverse por la pantalla y orientarse en el orden en el que el usuario hubiera seleccionado las acciones.

Para la rotación del objeto, se creó una clase de estilo por cada posible orientación (90°, 180° o 270°) las cuales rotaban la imagen ese ángulo respecto a la posición inicial. También se creó una variable posición para que el sistema tuviera una constancia de la orientación en la que se encontraba el personaje.

Generar el movimiento por cada acción era bastante sencillo, sin embargo, el procesamiento se hacía tan rápido que solo se era capaz de distinguir un movimiento linear desde el punto de inicio hasta el final. Para solventar este problema y producir un desplazamiento más lento y perceptible como en el juego original, al pulsar el botón azul, comenzaba un intervalo mediante la función “setInterval” la cual llamaba a la función que se le pasa como parámetro (“moves”) durante un tiempo indicado (1 segundo).

Dicha función “moves” busca el siguiente movimiento que se encuentre sin marcar como realizado. Para realizar este movimiento, la función tiene en cuenta la orientación en la que se encuentra el personaje y dependiendo de que movimiento se trata avanza hacia la dirección de la orientación del personaje o rota el personaje en la nueva dirección. En caso de cambiar de dirección también se actualiza la variable “posición”.

En la siguiente figura (ver Figura 59) podemos ver un fragmento de esta función en el caso de que la posición sea “arriba”.

```

case "arriba":
    if (document.getElementById("m" + i + j).src.indexOf("forward") > -1) {
        document.getElementById("m" + i + j).style = "opacity:0.5";
        document.getElementById("cubitto").style.top = auxX - cube + "px";
        auxX -= cube;
        x--;
    } else if (document.getElementById("m" + i + j).src.indexOf("left") > -1) {
        document.getElementById("m" + i + j).style = "opacity:0.5";
        document.getElementById("cubitto").classList.add('rotate270');
        posicion = "izquierda";
    } else if (document.getElementById("m" + i + j).src.indexOf("right") > -1) {
        document.getElementById("m" + i + j).style = "opacity:0.5";
        document.getElementById("cubitto").classList.add('rotate90');
        posicion = "derecha";
    }
    break;

```

Figura 59: Fragmento función moves

Si se ha procesado una acción, la variable “done” se pone a “true” lo que quiere decir que puede que todavía queden más acciones por realizar pero que ya se ha realizado una y, por tanto, hay que esperar al siguiente intervalo. En caso de que la variable “done” tenga valor “false” no se ha encontrado ninguna acción por realizar o se ha llegado al final, por lo que se para el intervalo y se comprueba si la posición alcanzada es la correcta mediante la función “isCorrect”.

4.4 Pruebas

Debido a la falta de tiempo, no ha sido posible un proceso de testeo de la aplicación con un gran número de usuarios. Se va a dividir este apartado en varios bloques, en el primer bloque se explicará las pruebas de control que he realizado sobre el funcionamiento de los distintos juegos y de la aplicación en sí. En el segundo bloque, se explicará las pruebas realizadas por varios usuarios.

4.4.1 Prueba General

Como se explicó en la figura 11, tras la implementación de cada minijuego, se comprobó el correcto funcionamiento de todos los elementos del minijuego.

Primero se comprobó el desarrollo normal del juego, es decir, se genera una variante, el usuario responde correctamente y sale del juego.

Tras esta comprobación se procedió a comprobar todos los caminos alternativos:

- El usuario decide seguir jugando tras contestar correctamente.
- El usuario contesta erróneamente y decide reintentarlo.
- El usuario decide salir al contestar erróneamente.
- El usuario decide salir antes de responder.
- El usuario pide ayuda antes de resolver.

Estas pruebas sirvieron para comprobar que todos los botones e interacciones funcionaban correctamente en especial las funciones y las ventanas modales.

Una vez comprobado esto, se repasó que el juego se viera correctamente en español y en inglés. Esta fue una parte importante donde se pudo resolver ciertos errores que aparecían como el cambio del tamaño de ciertos objetos debido a la mayor extensión de la letra.

Por último, se comprobó la correcta visualización del juego en distintos tamaños de pantalla, para comprobar que no se descuadrara demasiado.

Una vez comprobado el funcionamiento de los juegos por separado, se juntó todos los juegos y se volvió a probar la aplicación centrándose en los cambios de una ventana a otra.

4.4.2 Pruebas de usuarios

La aplicación ha sido probada por varios usuarios de distintas edades a fin de comprobar si era accesible por rangos de público distinto.

En primer lugar, la aplicación fue utilizada por un niño de 9 años. Aunque no pertenece al rango de edad para el que está diseñado el juego, se decidió que fuera el primer usuario para tener otro punto de vista sobre la aplicación de una edad más cercana al público objetivo.

Este usuario tardó muy poco tiempo en entender el funcionamiento de los juegos y casi no necesitó ayuda por parte de la aplicación. Una vez hubo entendido como funcionaban los juegos y tras acertar varias variables empezó a perder el interés.

El segundo usuario fue por un niño de 6 años. Este usuario sí que pertenece al rango de edad para el que está diseñado el juego, por lo que esta prueba era realmente importante.

Este usuario necesitó revisar las imágenes y videos de ayuda para comprender como funcionaban gran parte de los juegos. Cometió varios errores al principio, pero tuvo un rápido ritmo de aprendizaje y consiguió responder a los juegos correctamente con gran soltura.

El último usuario de prueba fue una niña de 3 años. Como hemos dicho anteriormente, este usuario necesita ayuda de un adulto para entender los juegos y no logró resolver algunos de los juegos que requieren más conocimientos como “Algoritmos”. Sin embargo, ayudada por el adulto fue capaz de resolver con soltura los juegos más fáciles como “Instrucciones” o “Cubitto”.

Con estas pruebas se pudo llegar a la conclusión de que la aplicación cumple con el objetivo para el que fue diseñado, aunque para un correcto aprendizaje y entendimiento el usuario debe saber leer y comprender las indicaciones de la aplicación.

5. CONCLUSIONES

El objetivo principal del proyecto consistía en la creación de una aplicación web educativa para el aprendizaje de los pensamientos computacionales en los más pequeños.

De esta propuesta podemos extraer dos objetivos principales: que permita el aprendizaje del pensamiento computacional y que sea fácilmente accesible para niños pequeños.

El aspecto de la aplicación está adecuado a niños pequeños mediante una tipografía infantil y fácil de leer. Además, la inclusión de personajes (figuras geométricas humanizadas) y los colores pastel, ayudan a que sea una aplicación atrayente para los niños.

Por otro lado, el juego permite el aprendizaje del pensamiento computacional mediante juegos sencillos e intuitivos. Estos juegos que pueden ser repetidos hasta dominar los conceptos o solo por diversión y enseñan a los usuarios métodos simples de resolución de problemas desde un punto de vista informático.

Por tanto, se puede llegar a la conclusión de que los objetivos han sido cumplidos. También cabe destacar que se han cumplido varios objetivos secundarios como la posibilidad de usar la aplicación en diferentes plataformas sin importar su tamaño de pantalla, la posibilidad de llevar el aprendizaje a cabo en español o en inglés o la subida del proyecto a un servidor lo que permite que pueda ser accedida desde el siguiente enlace: <http://www.lite.etsii.urjc.es/comphink/>.

Además, la realización de este proyecto ha permitido profundizar en conceptos relacionados con el desarrollo de aplicaciones web y al aprendizaje de lenguajes como PHP los cuales no se ven con detalle durante la formación universitaria.

Este proyecto ha supuesto un desarrollo personal debido a que ha permitido la creación de un proyecto desde sus inicios y la adquisición de conocimientos relacionados con el diseño y desarrollo de aplicaciones web del que no se disponía, en gran medida, al comienzo del proyecto. Se ha conseguido realizar una reingeniería de procesos, pues se ha reutilizado una aplicación ya existente para la realización del proyecto, aunque no se ha podido aprovechar parte del código, el diseño ha sido más sencillo por este motivo. Además, se ha producido un trabajo de ingeniería bastante extenso al tener que adaptar el juego Cubetto que era físico, a un juego online como Cubitto.

Por tanto, se puede concluir que el desarrollo de este proyecto ha tenido un gran éxito, tanto como por el cumplimiento de los objetivos que se presentaron inicialmente como por la experiencia y conocimientos adquiridos con la realización de este.

6. TRABAJOS FUTUROS

En este apartado se explicarán posibles mejoras y nuevas funcionalidades que se podrían implementar en un futuro a fin de mejorar la experiencia del usuario y la funcionalidad de la aplicación.

6.1 Conversión a Progressive Web App

Una progressive web app es una aplicación web que se puede descargar como aplicación nativa en un teléfono o Tablet. Esto es posible gracias a las nuevas características que ofrece HTML5 y a algunas API's propias del navegador. Además, esta característica permitiría el envío de notificaciones a los usuarios y el uso de la app sin conexión a internet (Google Developers, 2019).

Para convertir la aplicación en una progressive web app, se necesitan varios elementos:

- Diseño *responsive*: debe poder adaptarse a cualquier tamaño de pantalla.
- Service Worker: permite la ejecución en segundo plano y sin conexión de la aplicación.
- Manifest: controla la apariencia de la aplicación cuando se descarga como aplicación nativa.

La aplicación ya cuenta con el primer elemento de la lista, pero, por falta de tiempo, no se ha podido implementar los demás elementos correctamente.

6.2 Permitir la incorporación de nuevas variantes

Para hacer más interactivo el juego y mejorar la experiencia de usuario, se podría crear un apartado que permitiera al usuario crear nuevas variantes para los juegos que utilizan base de datos.

Esto permitiría que los niños con edades por encima del rango para el que está diseñado el juego o los padres de niños de edades recomendadas, crearan sus propias variantes personalizando así el juego. Además, así los niños podrían aplicar los conocimientos adquiridos tras usarlo.

6.3 Incluir niveles de dificultad

Para ampliar el rango de edad para el que la aplicación está diseñada, se podría implementar la diferenciación de variantes por niveles de dificultad. Para ello, habría que crear una ventana previa entre el menú principal y los juegos a fin de que el usuario eligiera la dificultad que prefiera, mostrándose esta elección como un rango de edad objetivo para dichas pruebas.

BIBLIOGRAFÍA

- The Apache Software Foundation. (2019). *NetBeans*. Obtenido de <https://netbeans.org/>
- Two Lives Left. (2019). *Two Lives Left*. Obtenido de <https://twolivesleft.com/CargoBot/>
- Acosta, A. S. (12 de Enero de 2017). *Programamos*. Recuperado el 10 de Junio de 2019, de <https://programamos.es/primeros-pasos-con-scratch-jr/>
- Adobe. (2019). *Adobe.com*. Obtenido de <https://www.adobe.com/es/products/xd.html>
- Apache Friends. (2019). *Apachefriends.org*. Recuperado el 5 de Junio de 2019, de <http://cort.as/xTYf>
- Bootstrap contributors. (2019). *Getbootstrap.com*. Obtenido de <https://getbootstrap.com/>
- Brackmann, C. (2017). Pensamiento Computacional Unplugged para niños: Enseñanza de la computación sin el protagonista.
- Cochran, D. (2012). *Twitter Bootstrap Web Development*. Packt Publishing.
- Common Sense Media. (2019). *Common Sense Media*. Obtenido de <https://www.commonsense.org/education/app/daisy-the-dinosaur>
- Fernández, J. I. (21 de Abril de 2014). *Programamos*. Recuperado el 10 de Junio de 2019, de <https://programamos.es/7-apps/>
- Fundación Montessori. (2018). *FAMM*. Recuperado el 18 de Junio de 2019, de <https://www.fundacionmontessori.org/metodo-montessori.htm>
- Gabriela Caguana Anzoategui, L. &. (2017). Cubetto for preschoolers: Computer programming code to code., (págs. 1-5). doi: 10.1109/SIIE.2017.8259649
- Gardey, J. P. (2010). *Definicion.de*. Recuperado el 7 de Junio de 2019, de <https://definicion.de/sql/>
- Google Developers. (2019). *Google Developers*. Recuperado el 17 de Junio de 2019, de <https://developers.google.com/web/progressive-web-apps/>
- Hopscotch Technologies. (2019). *hopscotch*. Obtenido de <https://www.gethopscotch.com/>
- i-Screammedia. (2017). *Truetruebot.com*. Recuperado el 10 de Junio de 2019, de <http://www.truetruebot.com/index/eng>
- Kodable. (2019). *Kodable*. Obtenido de <https://www.kodable.com/>
- Kusterer, R. (2019). *Netbeans.org*. Recuperado el 7 de Junio de 2019, de <https://netbeans.org/features/platform/index.html>
- Learning Resources. (2019). *Learning Resources*. Obtenido de <https://www.learningresources.com/product/learning+essentials--8482-+programmable+robot+mouse.do>

LightBot Inc. (2017). *LightBot*. Obtenido de <https://lightbot.com/>

Lopez, J. C. (11 de Agosto de 2017). *prezi.com*. Recuperado el 11 de Junio de 2019, de <http://cort.as/-JUJc>

MariaDB Foundation. (2019). *MariaDB.org*. Obtenido de <https://mariadb.org/about/>

MDN. (2019). *Documentación web de MDN*. Obtenido de <https://developer.mozilla.org/es/docs/HTML/HTML5>

MDN. (11 de Mayo de 2019). *MDN web docs*. Recuperado el 5 de Junio de 2019, de <http://cort.as/-8WOi>

OpenClassrooms. (30 de Octubre de 2017). *OpenClassrooms*. Recuperado el 11 de Junio de 2019, de <http://cort.as/-JUC5>

Perl.org. (2019). *Perl.org*. Obtenido de <https://www.perl.org/>

Primo Toys. (2019). *Primo Toys*. Obtenido de <https://www.primotoys.com/es/>

Robots educativos. (2019). *Robots para niños*. Recuperado el 10 de Junio de 2019, de <https://www.robotsparaninos.com/bee-bot-el-robot-abeja-2/>

Robots educativos. (2019). *Robots para niños*. Recuperado el 10 de Junio de 2019, de <https://www.robotsparaninos.com/code-and-go-robot-mouse-activity-set/>

ScratchJr. (2019). *ScratchJr*. Obtenido de <https://www.scratchjr.org/>

Systemax.jp. (2019). Recuperado el 10 de Junio de 2019, de <http://www.systemax.jp/en/sai/>

Terrapin. (2019). *Terrapinlogo.com*. Obtenido de <https://www.terrapinlogo.com/>

The PHP Group. (2019). *Php.net*. Recuperado el 7 de Junio de 2019, de <https://www.php.net/manual/es/intro-what-is.php>

TrueTrue. (2019). *Truetrue.es*. Obtenido de <http://www.truetrue.es/empezar.html>

Uniwebsidad. (2019). *Uniwebsidad*. Obtenido de <http://cort.as/-JEFD>

Wing, J. (Marzo de 2006). Computational Thinking. *CACM Viewpoint*, 33-35. Recuperado el 18 de Junio de 2019, de <http://cort.as/-JrAl>